



Cisco Webex Contact Center Desktop Developer Guide

First Published: 2020-12-02

Last Modified: 2021-01-22

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/c/en/us/about/legal/trademarks.html>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2021 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1

Introduction 1

Change History 1

Agent Desktop 2

Technical Overview 2

PART I

Widgets 3

CHAPTER 2

Custom Widget 5

Build a Custom Widget 5

Get Started 6

Define Property or Attribute Interface 6

Reactive Property or Attribute Change 7

Data Provider—Widget Properties and Attributes 8

Momentum UI Web Component Library 9

Light or Dark Mode Support 10

PART II

JavaScript SDK and Modules 13

CHAPTER 3

JavaScript SDK 15

JavaScript SDK 15

Get Started 16

Root JavaScript SDK Module 17

CHAPTER 4

Configuration Module 19

Methods 19

init(accessToken, Service) 19

[clientLocale\(\)](#) 20

CHAPTER 5 **Localization Module** 21

Methods 22

[init\(initOptions\)](#) 22

[createInstance\(createOptions\)](#) 23

[createMixin\(\)](#) 23

[getMergedInitOptions\(\)](#) 23

[cleanup\(\)](#) 24

CHAPTER 6 **Actions Module** 25

Methods 25

[getToken\(\)](#) 25

[getIdleCodes\(\)](#) 25

[getWrapUpCodes\(\)](#) 26

[getMediaTypeQueue\(telephony, social, email, chat\)](#) 26

[fireGeneralSilentNotification\(raw\)](#) 32

[fireGeneralAutoDismissNotification\(raw\)](#) 33

[fireGeneralAcknowledgeNotification\(raw\)](#) 34

[addCustomTask\(\)](#) 35

[getTaskMap\(\)](#) 37

CHAPTER 7 **Logger Module** 39

Methods 40

[createLogger\(my-custom-component\)](#) 40

[browserDownloadLogsJson\(my-custom-component\)](#) 40

[browserDownloadLogsText\(my-custom-component\)](#) 40

[getLogsCollection\(my-custom-component\)](#) 41

[getLogsJsonUrl\(my-custom-component\)](#) 41

[getLogsTextUrl\(my-custom-component\)](#) 41

[cleanupPrefixedLogs\(my-custom-component\)](#) 42

CHAPTER 8 **Agent State Information Module** 43

Methods 44

latestData	44
stateChange(state, auxCodeIdArray)	45
fetchAddressBooks()	45
Events	46
addEventListener	46

CHAPTER 9

Agent Contact Module 47

Methods	49
accept()	49
consultAccept()	53
end()	57
buddyAgents()	60
consult()	61
consultConference()	66
consultEnd()	70
decline()	73
cancelCtq()	74
wrapup()	76
vTeamList()	79
vTeamTransfer()	81
blindTransfer()	84
consultTransfer()	88
hold()	93
unHold()	95
pauseRecording()	98
resumeRecording()	101
Asynchronous Events	104
Methods	105
addEventListener(event, handler)	105
removeEventListener(event, handler)	105
addOnceEventListener(event, handler)	105
removeOnceEventListener(event, handler)	106
removeAllEventListeners(event, handler)	106
Events	106

eAgentContact	106
eAgentContactAssigned	108
eAgentContactAssignFailed	108
eAgentContactWrappedUp	108
eAgentOfferContact	108
eAgentOfferContactRona	111
eAgentOfferConsult	112
eAgentWrapup	114
eAgentContactHeld	114
eAgentContactUnHeld	114
eCallRecordingStarted	114
eResumeRecording	117
ePauseRecording	117
eConsultTransfer	117
eAgentblindTransferred	118
eAgentvteamTransfer	118
eAgentConsultCreated	118
eAgentConsultConferenced	118
eAgentConsultEnded	118
eAgentCtqCancelled	119
eAgentConsultConferenceEnded	119
eAgentConsulting	119
eAgentConsultFailed	119
eAgentConsultEndFailed	119
eAgentCtqFailed	119
eAgentCtqCancelFailed	120
eAgentConsultConferenceEndFailed	120

CHAPTER 10
Dialer Module 121**Methods 122**

startOutdial(data) 122

Events 124

eOutdialFailed 124

CHAPTER 11	Screen Pop Module	127
	Events	127
	eScreenPop	127

CHAPTER 12	Shortcut Key Module	129
	Methods	130
	register()	130
	getRegisteredKeys()	131
	Default Shortcut Keys	135
	Modifiers	138
	Registered Keys	139
	unregisterKeys()	139
	listenKeyPress(event)	140
	listenKeyConflict(event)	140
	listenConflictResolved()	141

PART III	Migration	143
-----------------	------------------	------------

CHAPTER 13	Finesse Gadget Migration	145
	Migrate Finesse Embedded Web Application Gadgets	145
	Migrate JavaScript Based Finesse Gadgets	147



CHAPTER 1

Introduction

- [Change History, on page 1](#)
- [Agent Desktop, on page 2](#)

Change History

This table lists the changes that are made to this guide. Most recent changes appear at the top.

Change	See	Date
<p>The JavaScript API reference is renamed to JavaScript SDK.</p> <p>Warning The references for the term <code>agentx</code> have been amended:</p> <ul style="list-style-type: none">• <code>@agentx/agentx</code> and <code>@agentx/agentx-services-types</code> will be exported under a common name <code>@wxcc-desktop/sdk-types</code>• <code>@agentx/agentx-js-api</code> is renamed to <code>@wxcc-desktop/sdk</code>• <code>agentxJsApi</code> is renamed to <code>Desktop</code>	-	January 2021
Added new modules, methods, and examples.	JavaScript SDK, on page 15	
Initial Release of this Document		December 2020

Agent Desktop

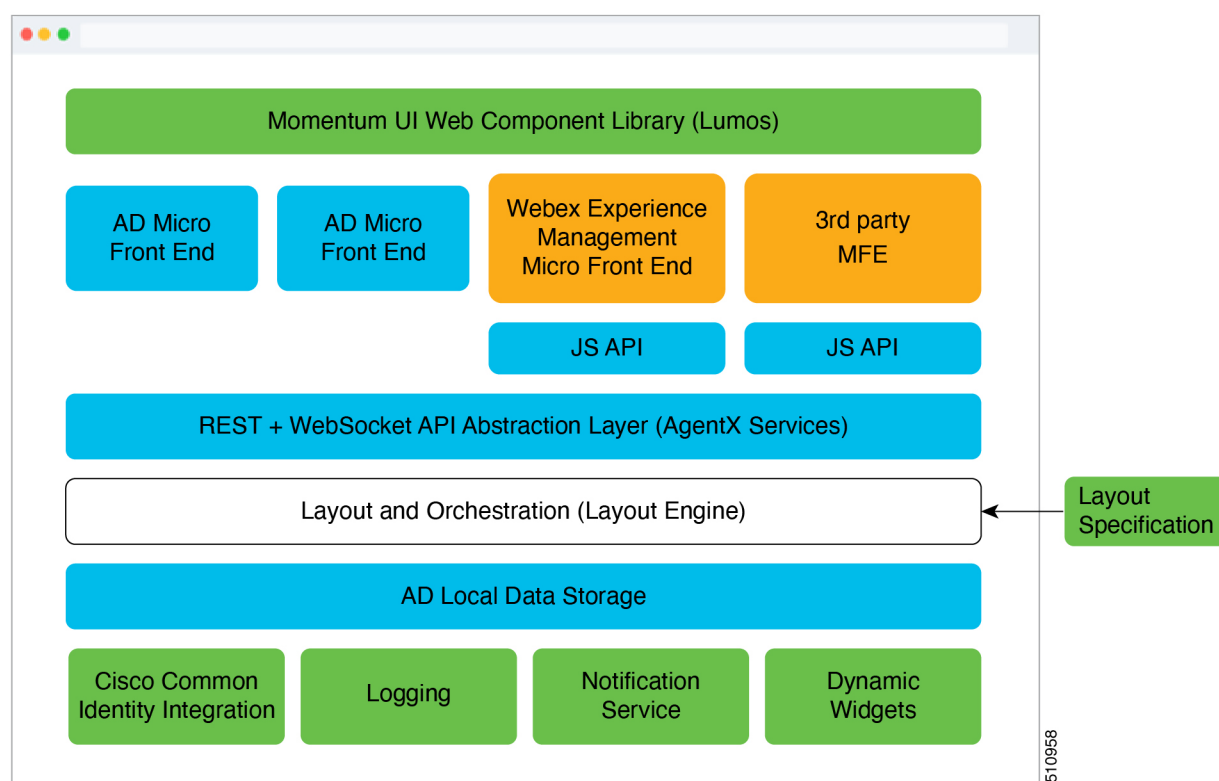
Cisco Webex Contact Center Agent Desktop brings your business the innovation, flexibility, and agility of cloud with the security and global scalability you have come to expect from Cisco. Keeping agents productive requires efficient processes and intuitive desktop tools. The more you can automate routine tasks, the more successful agents are at serving your customers.

Agent Desktop provides all your customer interactions—voice, chat, email, and social chat (Facebook Messenger and SMS) within one unified desktop experience. Application switching can be minimized with integrated CRM and other business applications.

Technical Overview

The following diagram shows the relationships between the various micro front ends and services.

Figure 1: Agent Desktop Architecture





PART I

Widgets

- [Custom Widget, on page 5](#)



CHAPTER 2

Custom Widget

- [Build a Custom Widget, on page 5](#)
- [Get Started , on page 6](#)
- [Define Property or Attribute Interface, on page 6](#)
- [Reactive Property or Attribute Change, on page 7](#)
- [Data Provider—Widget Properties and Attributes, on page 8](#)
- [Momentum UI Web Component Library, on page 9](#)

Build a Custom Widget

Cisco Webex Contact Center Agent Desktop supports extensions in the form of widgets. Widgets are essential components of desktop customization. A widget is a component with some specific encapsulated functionality, exported as a custom HTML element that is placed within the desktop. To be more precise, these custom HTML elements we recommend to also be Web Components, as any widget ends up being placed inside a nested tree of Shadow Roots, and will require a Shadow Document Object Model (DOM) of its own to be enabled for styling encapsulation (or any styling at all, for that matter).

Learning that any widget for Webex Contact Center Agent Desktop must be a Web Component or an embedded iFrame, a developer might feel forced into a technology. However, the developer can build these components on any compatible framework. The only requirement is that the custom widget must be exported as a Web Component. This is to ensure that any functionality is wrapped into a single custom HTML element that is defined within the JS bundle, with Shadow DOM enabled.

The following are a few examples:

- If you choose to build your widget in Angular JS, make sure that your Angular version supports [Angular Elements](#) (added in version 6), and your application is exported as an Angular Element.
- If you choose to build your application in React (similar to parts of Agent Desktop), you can wrap your React application into a custom HTML element and enable Shadow DOM manually, or use some of the open-source solutions that automate this process for you. For example, [Direflow](#).
- There are various other frameworks that are fully prepared to consume Web Components and be exported as Web Component. For reference, you can use <https://custom-elements-everywhere.com> that regularly runs automated tests against popular frameworks to check their ongoing compatibility.

You do not have to use a framework to build a web application to be exported as a widget. In fact, all examples in this documentation reference vanilla JavaScript.

Get Started

We suggest to have a setup in which you are able to place your widget in a larger Agent Desktop environment. For reference, you can use one of our boilerplate projects available at the Cisco DevNet Git Hub community: <https://github.com/CiscoDevNet/webex-contact-center-widget-starter>.

Alternatively, we suggest you to have a sandbox environment where you are able to place your widget (Web Component) in a constrained container and have an opportunity to test the following:

- Responsive behavior for various container sizes (from preferred to full screen).
- Overflow behavior on how your widget appears and behaves when its contents overflow on either X or Y axis (that is, scrollbars).
- Reacting to data passed into the widget. For more information about reactive component practices, see [Reactive Property or Attribute Change, on page 7](#).
- Light or Dark mode support whether you are using Momentum UI Web Component library or not.

Define Property or Attribute Interface

Prerequisite:

You must have an understanding of working with the Web Components and custom HTML elements. For more information, see [MDN documentation](#).

Your custom element may extend the HTMLElement class or the one that inherits from HTMLElement class.

Example:

```
class MyCustomElement extends HTMLElement {
  constructor() {
    // Always call super first in constructor
    super();
    const shadow = this.attachShadow({
      mode: 'open'
    });

    // write element functionality in here

    ...
  }
}
```

To define properties and attributes, assign the default values to them in the constructor.



Note

Any property can be passed as an attribute as long as it accepts data of the String or Boolean data types. For more information, see [JavaScript.Info](#).

Example:

```
class MyCustomElement extends HTMLElement {
  constructor() {
    // Always call super first in constructor
```

```
    super();
    const shadow = this.attachShadow({
      mode: 'open'
    });

    this.width = '100px';
    this.height = '100px';
    this.active = true;

    // Data of types Array, Object, Map, Set, etc. cannot be passed as an attribute
    this.options = [];
  }
}
```

Reactive Property or Attribute Change

To retrieve the latest data through attributes and properties from Agent Desktop (`widget.connectedCallback()`), rely on the `HTMLElement` lifecycle methods. For more information, see [MDN documentation](#).

**Note**

If lifecycle methods are not specified, your component will not react to new data that is being passed to it after the initial render.

Example:

```
class MyCustomElement extends HTMLElement {
  constructor() {
    // Always call super first in constructor
    super();
    const shadow = this.attachShadow({
      mode: 'open'
    });

    this.width = '100px';
    this.height = '100px';
    this.active = true;

    // Data of types Array, Object, Map, Set, etc. cannot be passed as an attribute
    this.options = [];
  }

  attributeChangedCallback(name, oldValue, newValue) {
    console.log('Custom element attributes changed.', name);
  }

  ...
}
```

Data Provider—Widget Properties and Attributes



Note

To receive real-time data through properties or attributes inside a custom widget, it has to be assigned appropriately in the layout JSON configuration by the administrator. For more information on JSON layout, see the *Desktop Layout* section in the *Provisioning* chapter of the [Cisco Webex Contact Center Setup and Administration Guide](#).

In addition to accessing data through JavaScript SDK subscribers, you can request data to be passed also through properties or attributes. If your component is built to react to changes in property or attribute, you will always get updated data this way from the single source of truth, the Agent Desktop. We call it a data provider.

Currently, we have a single data provider under the key STORE. Below, you will find a breakdown of all possible data and type definitions that is available through the STORE key:

Example: STORE Key Details

```
STORE.agent.agentId: string;
STORE.agent.agentName: string;
STORE.agent.agentPhoto: string;
STORE.agent.agentProfileID: string;
STORE.agent.allowConsultToQueue: boolean;
STORE.agent.dialPlan: Service.Aqm.Configs.DialPlan;
STORE.agent.dnNumber: string;
STORE.agent.enterpriseId: string;
STORE.agent.enterpriseId: string;
STORE.agent.cadVariables: Service.Aqm.Configs.CadVariables[];
STORE.agent.channels: {
    voiceCount: number,
    chatCount: number,
    emailCount: number,
    socialCount: number
};
STORE.agent.idleCodes: Service.Aqm.Configs.Entity[];
STORE.agent.idleStatusTimestamp: Date;
STORE.agent.isAgentAvailableAfterOutdial: boolean;
STORE.agent.isAdhocDialingEnabled: boolean;
STORE.agent.isCampaignManagementEnabled: boolean;
STORE.agent.isEndCallEnabled: boolean;
STORE.agent.isOutboundEnabledForAgent: boolean;
STORE.agent.isOutboundEnabledForTenant: boolean;
STORE.agent.privacyShieldVisible: string;
STORE.agent.profileType: string;
STORE.agent.subStatus: string;
STORE.agent.subStatusChangeTimestamp: Date;
STORE.agent.teamId: string;
STORE.agent.teamName: string;
STORE.agent.wrapUpData: Service.Aqm.Configs.WrapupData;
```

Example: Application Level

```
STORE.app.logo: string;
STORE.app.title: string;
STORE.app.darkMode: boolean;
```

Example: Single Sign-On (SSO)

```
STORE.auth.accessToken: string;
```


Example: Application Notifications

```
STORE.generalNotifications.isNotificationsEnabled: boolean;
STORE.generalNotifications.isSilentNotificationsEnabled: boolean;
STORE.generalNotifications.countActivated: number;
STORE.generalNotifications.countDeactivated: number;
STORE.generalNotifications.countPending: number;
STORE.generalNotifications.countAdded: number;
```

Example: JSON Layout

```
"widgets": {
  "my-component": {
    "comp": "my-custom-component",
    "properties": {
      "agentDnNumber": "$STORE.agent.dnNumber",
      "subStatus": "$STORE.agent.subStatus",
      "teamId": "$STORE.agent.teamId",
    }
  }
}
```

Momentum UI Web Component Library

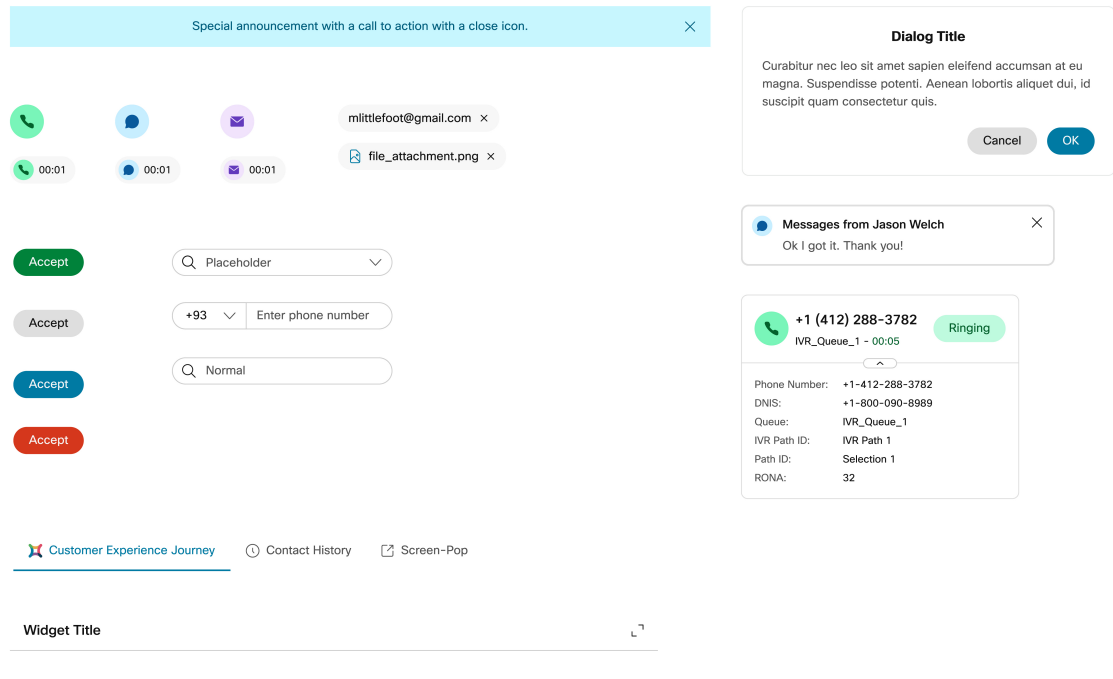
We encourage you to explore our UI component library to enable you to build a custom widget interface faster with the Webex Contact Center Agent Desktop visual language. For more information on the momentum UI Web Component library, see [GitHub](#) and [MIT](#) license. If you encounter any issues, report them on our [GitHub](#) page.

The current collection includes:

- Activity Button
- Alert
- Alert Banner
- Avatar
- Badge
- Breadcrumbs
- Button
- Checkbox
- Editable Field
- Floating Modal
- Help Text
- Icon
- Input
- Label
- Link
- List
- List Item
- Loading indicator
- Meeting Alert
- Menu Overlay
- Modal
- Phone Number Input
- Progress Bar
- Radio
- Spinner
- Table
- Tabs
- Task Item
- Theme
- Toggle Switch
- Tooltip

Momentum UI Web Component library will continue to grow and improve over time. We welcome any contributions. For more information on the Web Components library, see [Components](#).

Figure 2: Sample Components on Agent



Light or Dark Mode Support

In Webex Contact Center Agent Desktop, we support Light and Dark mode by default. This is achieved through Momentum UI Web Component `<md-theme>`. When wrapped around components, `<md-theme>` assigns values to a collection of predefined core CSS color variables and [custom CSS properties](#). These properties switch from light to dark on a theme switch. The advantage of relying on CSS custom properties is that they pierce Shadow DOM and can be used inside many nested layers of Shadow DOM.



Note

You can use the core CSS color variables if your widget CSS is in the following format:

```
.container {
  background-color: var(--md-primary-bg-color, #fff);
}
```

Example: Light Mode

```
--md-default-focus-outline-color: #{$md-blue-60};

--md-primary-bg-color: #{$md-white};
--md-secondary-bg-color: #{$md-gray-05};
--md-secondary-white-bg-color: #{$md-white};
--md-tertiary-bg-color: #{$md-gray-10};
--md-tertiary-white-bg-color: #{$md-white};
--md-quaternary-bg-color: #{$md-gray-20};

--md-primary-success-bg-color: #{$md-green-10};
--md-primary-success-text-color: #{$md-green-70};

--md-primary-text-color: #{$md-gray-100};
--md-secondary-text-color: #{$md-gray-70};
```

```
--md-disabled-text-color: #{$md-gray-40};
--md-highlight-text-color: #{$md-theme-20};
--md-hyperlink-text-color: #{$md-theme-70};
--md-hyperlink-hover-text-color: #{$md-theme-90};
--md-hyperlink-focus-text-color: #{$md-theme-70};

--md-primary-seperator-color: #{$md-gray-30};
--md-secondary-seperator-color: #{$md-gray-40};

--md-presence-active-bg-color: #{$md-green-50};
--md-presence-do-not-disturb-bg-color: #{$md-red-60};
--md-presence-away-bg-color: #{$md-gray-30};
--md-presence-busy-bg-color: #{$md-yellow-40};
```

Example: Dark Mode

```
--md-default-focus-outline-color: #{$md-blue-40};

--md-primary-bg-color: #{$md-gray-100};
--md-secondary-bg-color: #{$md-gray-95};
--md-secondary-white-bg-color: #{$md-gray-95};
--md-tertiary-bg-color: #{$md-gray-90};
--md-tertiary-white-bg-color: #{$md-gray-90};
--md-quaternary-bg-color: #{$md-gray-80};

--md-primary-success-bg-color: #{$md-mint-70};
--md-primary-success-text-color: #{$md-mint-20};

--md-primary-text-color: #{$md-gray-05};
--md-secondary-text-color: #{$md-gray-40};
--md-disabled-text-color: #{$md-gray-70};
--md-highlight-text-color: #{$md-theme-80};
--md-hyperlink-text-color: #{$md-theme-40};
--md-hyperlink-hover-text-color: #{$md-theme-20};
--md-hyperlink-focus-text-color: #{$md-theme-40};

--md-primary-seperator-color: #{$md-gray-70};
--md-secondary-seperator-color: #{$md-gray-60};

--md-presence-active-bg-color: #{$md-green-50};
--md-presence-do-not-disturb-bg-color: #{$md-red-60};
--md-presence-away-bg-color: #{$md-gray-30};
--md-presence-busy-bg-color: #{$md-yellow-40};

--md-auto-wrapup-bg-color: #{$md-blue-90};
```




PART II

JavaScript SDK and Modules

- [JavaScript SDK](#), on page 15
- [Configuration Module](#), on page 19
- [Localization Module](#), on page 21
- [Actions Module](#), on page 25
- [Logger Module](#), on page 39
- [Agent State Information Module](#), on page 43
- [Agent Contact Module](#), on page 47
- [Dialer Module](#), on page 121
- [Screen Pop Module](#), on page 127
- [Shortcut Key Module](#), on page 129



CHAPTER 3

JavaScript SDK



Warning

Restrain from using the old libraries such as `@agentxJsApi`, `@agentx/agentx`, `@agentx/agentx-services-types` and `@agentx/agentx-js-api`.

The references for the term `agentx` have been amended:

- `@agentx/agentx` and `@agentx/agentx-services-types` are exported under a common name `@wxcc-desktop/sdk-types`
- `@agentx/agentx-js-api` is renamed to `@wxcc-desktop/sdk`
- `agentxJsApi` is renamed to `Desktop`

For more information on using the JavaScript SDK, see [Get Started](#).

- [JavaScript SDK, on page 15](#)
- [Root JavaScript SDK Module, on page 17](#)

JavaScript SDK

The Agent Desktop JavaScript SDK is an npm package that allows you to request up-to-date information from the Agent Desktop. Using the SDK, you can request information such as agent details, assigned tasks, details of specific tasks, current browser locale, and the authentication token for Single Sign-On (SSO) integration.

The SDK package allows you to:

- request data to be passed to your widgets through properties and attributes.
- perform more complex operations by consuming and manipulating the system data inside your widget.
- subscribe to data arriving asynchronously.

Some events in the Agent Desktop occur asynchronously. To subscribe to asynchronous events and access data within the payload, you can add a listener. A few examples for asynchronous events are:

- New task offered
- New task assigned

- Consult request created
- Consult ended
- Screen pop arrived

For the complete list of asynchronous events, see [Asynchronous Events](#).

Get Started

To start using the JavaScript SDK, you can use any of the following options:

- Run the following command in your project folder:

```
npm install @wxc-desktop/sdk --save
```

or

```
yarn add @wxc-desktop/sdk
```

- Run the following command to add a package to your `package.json` file:

```
"dependencies": {
  "@wxc-desktop/sdk": "^1.2.2"
},
```

- If you also choose to use our Momentum-ui Web Component library, you must add:

```
"peerDependencies": {
  "@momentum-ui/core": "19.9.2",
  "@momentum-ui/icons": "7.45.0",
  "@momentum-ui/utils": "6.2.7",
  "@momentum-ui/web-components": "^2.0.13",
  "lit-element": "^2.3.1",
  "lit-html": "^1.2.1"
},
```

In addition, to access the type definition of return Promise for the JavaScript SDK requests, install the following package:

```
"devDependencies": {
  "@wxc-desktop/sdk-types": "^1.0.2",
  ...
},
```



Important

Momentum and lit-element dependencies are added to `peerDependencies`. These dependencies are present in Agent Desktop, and should not be imported twice. There is no way to maintain the same versions in your widget and in Agent Desktop.

Once you have installed the package in your project, include it in the appropriate component file following the ES6 import pattern:

```
import {Desktop} from "@wxc-desktop/sdk";
```


Root JavaScript SDK Module

`Desktop` is the root module of the JavaScript SDK. The root module provides a reference to the following sub-modules:

- [Configuration Module, on page 19](#)
- [Localization Module, on page 21](#)
- [Actions Module, on page 25](#)
- [Logger Module, on page 39](#)
- [Agent State Information Module, on page 43](#)
- [Agent Contact Module, on page 47](#)
- [Dialer Module, on page 121](#)
- [Screen Pop Module, on page 127](#)
- [Shortcut Key Module, on page 129](#)

Example

```
import {
  Desktop
} from "@wxcc-desktop/sdk";

const {
  config,
  i18n,
  actions,
  agentContact,
  agentStateInfo,
  dialer,
  logger,
  screenpop,
  shortcutKey,
} = Desktop;
```




CHAPTER 4

Configuration Module

The `Desktop.config` module provides configuration details of Agent Desktop.

Example

```
import {
    Desktop
} from "@wxcc-desktop/sdk";
import {
    SERVICE
} from "@wxcc-desktop/sdk-types";

await Desktop.config.init();

// After Desktop config initied, all sub-modules will inject SERVICE instance via their
init() methods automatically

// CLEANUP Desktop config is possible to re-use modules with SERVICE configured to another
environment
Desktop.config.cleanup();

// After Desktop config cleaned, all sub-modules will cleanup themselves via their cleanup()
methods automatically
```

- [Methods, on page 19](#)

Methods

init(accessToken, Service)

Initiates the configuration module to start utilizing any of the `Desktop` modules.

Example

```
connectedCallback() {
    super.connectedCallback();
    this.init(); // Initialization the configuration without any parameters optional
parameters are { accessToken: ACCESS_TOKEN, SERVICE: SERVICE }
}

async init() {
    await Desktop.config.init()
}
```

For more information on the widget starter example, see [Cisco Webex Contact Center Widget Starter](#).

Parameters

Name	Type	Description	Required
accessToken	String	Clients are issued an access token that contains identity information for the user that is encrypted by default. It accesses protected resources.	Optional
Service	String	Services provided by the Desktop modules such as localization, actions, logger, and so on.	Optional

clientLocale()

Requests system data such as locale of the client.

```
const locale = Desktop.config.clientLocale;
```

Returns

{String} The locale of the client.

Example Response

```
const clientLocaleResponse = "en-US"
```



CHAPTER 5

Localization Module

The `Desktop.i18n` module creates and maintains the localization bundles for lit-element based widgets (in case you considered [Starter Widget](#) as a base).

The localization module is built based on the [i18next](#) package and enables the widget developer to utilize the Agent Desktop internationalization mechanism and load additional localization bundles to it. For more information on `i18next`, refer the following resources:

- For `Desktop.i18n` instantiating object, see <https://www.i18next.com/overview/api#instance-creation>.
- For `i18n` instance back-end configuration, see <https://github.com/i18next/i18next-http-backend>.
- For `i18n` instance `languageDetector` configuration, see <https://github.com/i18next/i18next-browser-languageDetector>.
- For `i18n` instance init options, see <https://www.i18next.com/overview/configuration-options>.

Using this module implies that your widget has `lit-element` and `lit-html` libraries.

```
import {
  Desktop
} from "@wcc-desktop/sdk";

...
// All CreateOptions for i18n are optional
type CreateOptions = {
  backend ? : Backend // import Backend from "i18next-http-backend";
  languageDetector ? : LanguageDetector // import LanguageDetector from
  "i18next-browser-languagedetector";
};

const i18n = Desktop.i18n.createInstance(createOptions ? : CreateOptions) // returns instance
  described in https://www.i18next.com/overview/api#instance-creation
const i18nMixin = Desktop.i18n.createMixin({
  i18n /*Injecting i18n service instance into lit-element mixin */
})

// FYI you can see default options like so
console.log(Desktop.i18n.DEFAULT_INIT_OPTIONS); // => i18n.init options that are using by
Desktop by default

// To get started, Init i18n with options to be able call "t" function translations
if (!i18n.isInitialized) {
  // Here, you are adding (merging) your localization package with the Agent Desktop
  existing set of packages
  const initOptions = Desktop.i18n.getMergedInitOptions(Desktop.i18n.DEFAULT_INIT_OPTIONS
  || {}, {
```

```

        defaultNS: "my-ns", // "ns" here stands for the default JSON file name containing
the localization
        ns: ["my-ns"],
        fallbackLng: "en",
        backend: {
            loadPath: "/.../path-to-locales/.../{lng}/{ns}.json"
        }
    });

    i18n.init(initOptions).catch(err => console.log(err));
}

```

When the service is initialized, create components with the mixing you created earlier:

```

import {
    customElement,
    LitElement
} from "lit-element";
import {
    html
} from "lit-html";

@customElement("my-awesome-component")
export class MyAwesomeComponent extends i18nMixin(LitElement) {
    render() {
        return html `
            <!-- i18nMixin will subscribe component tree updates on languages load & language
change -->
            <!-- Component wrapped by i18nMixin can access t funcation via this.t(...) -->
            <p>${this.t("my-ns:key1")}</p>` <
                p > $ {
                    this.t("my-ns:key2")
                } < /p>`
        }
    }
}

```

- [Methods, on page 22](#)

Methods

init(initOptions)

Initiates the localization module.

Example

```
i18n.init(initOptions)
```

Parameters

Name	Type	Description	Required
initOptions	String	The init options that are used by Desktop by default. <code>console.log(Desktop.i18n.DEFAULT_INIT_OPTIONS)</code>	Yes

createInstance(createOptions)

Creates the localization bundles for lit-element based widgets.

Example

```
const i18n = Desktop.i18n.createInstance(createOptions ? : CreateOptions)
// returns instance described in https://www.i18next.com/overview/api#instance-creation
```

Parameters

Name	Type	Description	Required
createOptions	String	Creates options for localization (i18n) module.	Yes

createMixin()

Creates the localization (i18n) service instance into lit-element mixing.

Example

```
const i18nMixin = Desktop.i18n.createMixin({
  i18n /*Injecting i18n service instance into lit-element mixin */
})
```

getMergedInitOptions()

Adds or merges your localization package with the existing set of packages in Agent Desktop.

Example

```
const initOptions = Desktop.i18n.getMergedInitOptions(Desktop.i18n.DEFAULT_INIT_OPTIONS ||
  {}, {
    defaultNS: "my-ns", // "ns" here stands for the default JSON file name containing the
    localization
    ns: ["my-ns"],
    fallbackLng: "en",
    backend: {
      loadPath: "/.../path-to-locales/.../{lng}/{ns}.json"
    }
  });
```

The following table lists the payload details:

Name	Type	Description	Required
DEFAULT_INIT_OPTIONS	String	The init options that are used by Desktop by default.	Yes
defaultNS	String	Default localization JSON file name.	Yes
ns	String	JSON file name of your locale.	Yes
fallbackLng	String	Language to use if the file is not found in the configured path or if you cannot provide the preferred language for a user.	Yes
backend	String	Contains the load path.	Yes

Name	Type	Description	Required
-->loadPath	String	Location of the file path.	Yes

cleanup()

Triggers the clean up when init service returns an undefined response.

Example

```
cleanup() {  
    this.SERVICE = undefined;  
  
    this.logger.info("Cleaned");  
}
```




CHAPTER 6

Actions Module

The `Desktop.actions` module retrieves real-time data from the client-side data store on Agent Desktop side.



Note

You can also pass the same information to your widget through properties, using our Data Provider. For more information, see [Data Provider—Widget Properties and Attributes](#).

- [Methods, on page 25](#)

Methods

getToken()

Retrieves the authentication token used for SSO authentication.

Example

```
const accessToken = await Desktop.actions.getToken()  
// => Get current accessToken from Desktop store
```

Returns

`{String}` A user access token.

getIdleCodes()

Retrieves the current idle codes. Example: Idle, Coffee break, Meeting, Tea.

Example

```
const idleCodes = await Desktop.actions.getIdleCodes()  
// => Get current idleCodes from Desktop store
```

Returns

`{Array}` The array of objects.

Example Response

```
{
  id: "1643",
  isDefault: false,
  isSystem: true,
  name: "RONA"
} {
  id: "1644",
  isDefault: true,
  isSystem: false,
  name: "Meeting"
}
```

getWrapUpCodes()

Retrieves the wrap up codes. Example: Credit Card Issue, Medical Query, Sales Explained.

Example

```
const wrapUpCodes = await Desktop.actions.getWrapUpCodes()
// => Get current wrapUpCodes from Desktop store
Retrieves wrap up codes such as Ex. "Credit Card Issue", "Medical Query", "Sales Explained"
as per its configured
```

Returns

{Array} The array of objects.

Example Response

```
{
  id: "2063",
  isDefault: false,
  isSystem: false,
  name: "Account Information Explained"
} {
  id: "2061",
  isDefault: false,
  isSystem: false,
  name: "Credit Card Issue"
}
```

getMediaTypeQueue(telephony, social, email, chat)

Retrieves the list of currently available media types.



Note

The administrator configures the media type for an agent.

Example

```
const queue = await Desktop.actions.getMediaTypeQueue("telephony" | "social" | "email" |
"chat") // => Get current media queue from Agent Desktop store
```

Parameters

Name	Type	Description	Required
telephony	String	The media channel type for voice calls.	Yes

Name	Type	Description	Required
social	String	The media channel type for social messaging conversations. The supported social messaging conversations are: <ul style="list-style-type: none"> • Messenger (Facebook Messenger) • SMS (Short Message Service) 	Yes
email	String	The media channel type for emails.	Yes
chat	String	The media channel type for chats.	Yes

Returns

{Object} The object with the retrieved data.

Example Response

```
{
  "key": "c9897fa7-6188-11eb-a6e4-f93bdeada4e5",
  "value": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "consultMediaResourceId": null,
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "accountId": {
          "agentEditable": false,
          "displayName": "accountId",
          "name": "accountId",
          "type": "STRING",
          "value": "e9e2c7a0-5c64-11ea-9e59-6fbf992ffc23"
        },
        "ani": {
          "agentEditable": false,
          "displayName": "ani",
          "name": "ani",
          "type": "STRING",
          "value": "janedoe@gmail.com"
        },
        "bccAddress": {
          "agentEditable": false,
          "displayName": "bccAddress",
          "name": "bccAddress",
          "type": "STRING",
          "value": ""
        },
        "ccAddress": {
          "agentEditable": false,
          "displayName": "ccAddress",
          "name": "ccAddress",
          "type": "STRING",
          "value": ""
        },
        "contentType": {
          "agentEditable": false,
          "displayName": "contentType",
          "name": "contentType",
          "type": "STRING",
          "value": "multipart/alternative"
        }
      }
    }
  }
}
```

```

    },
    "customerName": {
      "agentEditable": false,
      "displayName": "customerName",
      "name": "customerName",
      "type": "STRING",
      "value": "Jane Doe"
    },
    "date": {
      "agentEditable": false,
      "displayName": "date",
      "name": "date",
      "type": "STRING",
      "value": "Wed, 20 Jan 2021 06:09:06 +0000"
    },
    "dn": {
      "agentEditable": false,
      "displayName": "dn",
      "name": "dn",
      "type": "STRING",
      "value": "agent2021@gmail.com"
    },
    "entryPointId": {
      "agentEditable": false,
      "displayName": "entryPointId",
      "name": "entryPointId",
      "type": "STRING",
      "value": "AXCZuH9MXrf9I0XFBIfL"
    },
    "from": {
      "agentEditable": false,
      "displayName": "from",
      "name": "from",
      "type": "STRING",
      "value": "Jane Doe"
    },
    "fromAddress": {
      "agentEditable": false,
      "displayName": "fromAddress",
      "name": "fromAddress",
      "type": "STRING",
      "value": "janedoe@gmail.com"
    },
    "inReplyTo": {
      "agentEditable": false,
      "displayName": "inReplyTo",
      "name": "inReplyTo",
      "type": "STRING",
      "value": ""
    },
    "messageId": {
      "agentEditable": false,
      "displayName": "messageId",
      "name": "messageId",
      "type": "STRING",
      "value": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com"
    },
    "queueType": {
      "agentEditable": false,
      "displayName": "queueType",
      "name": "queueType",
      "type": "STRING",
      "value": "tam"
    },
  },

```

```

    "reasonCode": {
      "agentEditable": false,
      "displayName": "reasonCode",
      "name": "reasonCode",
      "type": "STRING",
      "value": "Email_Queue"
    },
    "references": {
      "agentEditable": false,
      "displayName": "references",
      "name": "references",
      "type": "STRING",
      "value": ""
    },
    "replyToAddress": {
      "agentEditable": false,
      "displayName": "replyToAddress",
      "name": "replyToAddress",
      "type": "STRING",
      "value": "janedoe@gmail.com"
    },
    "ronaTimeout": {
      "agentEditable": false,
      "displayName": "ronaTimeout",
      "name": "ronaTimeout",
      "type": "STRING",
      "value": "30"
    },
    "subject": {
      "agentEditable": false,
      "displayName": "subject",
      "name": "subject",
      "type": "STRING",
      "value": "sales"
    },
    "threadId": {
      "agentEditable": false,
      "displayName": "threadId",
      "name": "threadId",
      "type": "STRING",
      "value": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com"
    },
    "toAddress": {
      "agentEditable": false,
      "displayName": "toAddress",
      "name": "toAddress",
      "type": "STRING",
      "value": "agent2021@gmail.com"
    },
    "virtualTeamName": {
      "agentEditable": false,
      "displayName": "virtualTeamName",
      "name": "virtualTeamName",
      "type": "STRING",
      "value": "Email_Queue"
    }
  },
  "callAssociatedDetails": {
    "accountId": "e9e2c7a0-5c64-11ea-9e59-6fbf992ffc23",
    "ani": "janedoe@gmail.com",
    "bccAddress": "",
    "ccAddress": "",
    "contentType": "multipart/alternative",
    "customerName": "Jane Doe",

```

```

    "date": "Wed, 20 Jan 2021 06:09:06 +0000",
    "dn": "agent2021@gmail.com",
    "entryPointId": "AXCZuH9MXrf9I0XFBIfL",
    "from": "Jane Doe",
    "fromAddress": "janedoe@gmail.com",
    "inReplyTo": "",
    "messageId": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com",
    "queueType": "tam",
    "reasonCode": "Email_Queue",
    "references": "",
    "replyToAddress": "janedoe@gmail.com",
    "ronaTimeout": "30",
    "subject": "sales",
    "threadId": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com",
    "toAddress": "agent2021@gmail.com",
    "virtualTeamName": "Email_Queue"
  },
  "callFlowParams": {
    "Automation_Email_Queue": {
      "description": "",
      "name": "Automation_Email_Queue",
      "qualifier": "vteam",
      "value": "3281",
      "valueDataType": "string"
    },
    "Dont_Use_Email": {
      "description": "",
      "name": "Dont_Use_Email",
      "qualifier": "vteam",
      "value": "3276",
      "valueDataType": "string"
    },
    "Email_Queue": {
      "description": "",
      "name": "Email_Queue",
      "qualifier": "vteam",
      "value": "3266",
      "valueDataType": "string"
    },
    "Mail_Automation_Queue": {
      "description": "",
      "name": "Mail_Automation_Queue",
      "qualifier": "vteam",
      "value": "3432",
      "valueDataType": "string"
    }
  },
  "callProcessingDetails": {
    "QMGrName": "aqm",
    "QueueId": "3266",
    "accountId": "e9e2c7a0-5c64-11ea-9e59-6fbf992ffc23",
    "ani": "janedoe@gmail.com",
    "bccAddress": "",
    "ccAddress": "",
    "contentType": "multipart/alternative",
    "customerName": "Jane Doe",
    "date": "Wed, 20 Jan 2021 06:09:06 +0000",
    "dnis": "agent2021@gmail.com",
    "entryPointId": "AXCZuH9MXrf9I0XFBIfL",
    "from": "Jane Doe",
    "fromAddress": "janedoe@gmail.com",
    "inReplyTo": "",
    "messageId": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com",
    "pauseDuration": "10",

```

```

    "pauseResumeEnabled": "true",
    "queueType": "tam",
    "reasonCode": "Email_Queue",
    "references": "",
    "replyToAddress": "janedoe@gmail.com",
    "ronaTimeout": "30",
    "subject": "sales",
    "taskToBeSelfServiced": "false",
    "tenantId": "133",
    "threadId": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com",
    "toAddress": "agent2021@gmail.com",
    "virtualTeamName": "Email_Queue",
    "vteamId": "AXCZuH9MXrf9I0XFBIfL"
  },
  "contactDirection": {
    "type": "INBOUND"
  },
  "currentVTeam": "3266",
  "interactionId": "c9897fa7-6188-11eb-a6e4-f93bdeada4e5",
  "isFcManaged": false,
  "isTerminated": false,
  "media": {
    "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com": {
      "holdTimestamp": null,
      "isHold": false,
      "mType": "mainCall",
      "mediaMgr": "emm",
      "mediaResourceId": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com",
      "mediaType": "email",
      "participants": [
        "janedoe@gmail.com",
        "7c867aa9-ec768-341a-b767-e5hd6ae7g701"
      ]
    }
  },
  "mediaChannel": "email",
  "mediaType": "email",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "outboundType": null,
  "owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
  "participants": {
    "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
      "channelId": "a72a75f8-fd30-44f9-85f1-39d633055475",
      "consultState": null,
      "consultTimestamp": null,
      "dn": "8895579172",
      "hasJoined": true,
      "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
      "isConsulted": false,
      "isWrapUp": false,
      "joinTimestamp": 1612248681335,
      "lastUpdated": 1612248681336,
      "name": "John Doe",
      "pType": "Agent",
      "queueId": "3266",
      "queueMgrId": "aqm",
      "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
      "siteId": "472",
      "teamId": "960",
      "teamName": "Email_Team",
      "type": "Agent",
      "wrapUpTimestamp": null
    },
    "janedoe@gmail.com": {

```

```

        "id": "janedoe@gmail.com",
        "pType": "Customer",
        "type": "Customer"
      }
    },
    "previousVTeams": [],
    "state": "connected",
    "workflowManager": null
  },
  "interactionId": "c9897fa7-6188-11eb-a6e4-f93bdeada4e5",
  "isConsulted": false,
  "mediaResourceId": "289DBEAD-0715-4878-A0B9-365555C18E72@cisco.com",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "2e30a830-6611-11eb-bbe4-81e078594d7a",
  "type": "AgentContact"
}
}

```

fireGeneralSilentNotification(raw)

Triggers silent notifications. Silent notifications do not appear on the desktop, but are listed in the Notification Center. The **Notification Center** icon indicates the increase in the unread message count.

Example

```

import {
  Desktop
} from "@wxcc-desktop/sdk";
import {
  Notifications
} from "@uuip/unified-ui-platform-sdk";

...

const raw: Notifications.ItemMeta.Raw = {
  data: {
    type: Notifications.ItemMeta.Type.Info,
    mode: Notifications.ItemMeta.Mode.Silent, // Change type here based on the method.
    title: "Info - Silent",
    data: "Lorem Ipsum Dolor",
  },
};

// Agent Desktop General Notifications:
Desktop.actions.fireGeneralSilentNotification(raw)
// => Fires silent notification in Agent Desktop.
// Silent notification will not have any apperance on desktop but a notification icon will
// have one count increased.

```

Parameters

Name	Type	Description	Required
raw	Object	Contains information regarding the type of notification sent to the user.	Yes
-->data	Object	Options of the specific notification.	Yes

Name	Type	Description	Required
--> type	Enum	The type of notifications displayed on the Agent Desktop. <ul style="list-style-type: none"> • Info • Warn • Error • Success • Chat • Default 	Yes
--> mode	Enum	The available mode of notification based on the method. <ul style="list-style-type: none"> • Silent • AutoDismiss • Acknowledge 	Yes
--> title	String	The title of the notification.	Yes
--> data	Object	The content of the notification.	Yes

fireGeneralAutoDismissNotification(raw)

Triggers the auto-dismiss notification. Returns the notification resolved status, reason, and mode.



Note If the Agent Desktop notifications are disabled by the user, they are converted into silent notifications and reflected in mode.

Example

```
import {
  Desktop
} from "@wxcc-desktop/sdk";
import {
  Notifications
} from "@uui/unified-ui-platform-sdk";

...

const raw: Notifications.ItemMeta.Raw = {
  data: {
    type: Notifications.ItemMeta.Type.Info,
    mode: Notifications.ItemMeta.Mode.AutoDismiss, // Change type here based on the
method.
    title: "Info - AutoDismiss",
    data: "Lorem Ipsum Dolor",
  },
};
```

```
// Unlike silent notification, auto-dismiss and acknowledge can have controlled responses,
// that may reflect in the notification status, e.g.:

// => Fires auto-dismiss notification in Agent Desktop. Returns notification resolved status,
// reason, mode.
NOTE: if Agent Desktop notifications are disabled by the user - it will be converted into
silent notification and reflected in "mode"
const [ status, reason, mode ]: [
  Notifications.ItemMeta.Status, Notifications.ItemMeta.StatusChangeEventReason,
  Notifications.ItemMeta.Mode ] = await Desktop.actions.fireGeneralAutoDismissNotification(raw)
```

Parameters

Name	Type	Description	Required
raw	Object	Contains information regarding the type of notification sent to the user. For more information, see the <i>Parameters</i> table of the fireGeneralSilentNotification(raw) method.	Yes

fireGeneralAcknowledgeNotification(raw)

Triggers the acknowledge notification. Returns the notification resolved status, reason, and mode.



Note

If the Agent Desktop notifications are disabled by the user, they are converted into silent notifications and reflected in mode.

Example

```
import {
  Desktop
} from "@wxcc-desktop/sdk";
import {
  Notifications
} from "@uuiplatform/unified-ui-platform-sdk";

...

const raw: Notifications.ItemMeta.Raw = {
  data: {
    type: Notifications.ItemMeta.Type.Info,
    mode: Notifications.ItemMeta.Mode.Acknowledge, // Change type here based on the
method.
    title: "Info - Acknowledge",
    data: "Lorem Ipsum Dolor",
  },
};

// => Fires acknowledge notification in Agent Desktop. Returns notification resolved status,
// reason, mode.
NOTE: if Agent Desktop notifications are disabled by the user - it will be converted into
silent notification and reflected in "mode"
const [ status, reason, mode ]: [ Notifications.ItemMeta.Status,
  Notifications.ItemMeta.StatusChangeEventReason, Notifications.ItemMeta.Mode ] = await
Desktop.actions.fireGeneralAcknowledgeNotification(raw)
```

Parameters

Name	Type	Description	Required
raw	Object	Contains information regarding the type of notification sent to the user. For more information, see the <i>Parameters</i> table of the fireGeneralSilentNotification(raw) method.	Yes

addCustomTask()

Adds a new custom task within your widget. The task appears in the Agent Desktop task list.

**Note**

To keep the list of tasks up to date within your widget, you must add event listeners and subscribe to asynchronous events. The events signify new tasks being assigned to an agent. For more information, see [Asynchronous Events](#).

Example

```
Desktop.actions.addCustomTask({
  export const contactPayload = {
    mediaResourceId: "49bcf26b-ec75-4351-89fa-55d54682c20c",
    eventType: "RoutingMessage",
    agentId: "16839506-7c48-4a71-ba1b-d585e5d37607",
    trackingId: "354deae-c98a-4a8c-9e04-e56e129fdb9f",
    interaction: {
      isFcManaged: false,
      isTerminated: false,
      mediaType: "telephony",
      previousVTeams: ["AXCZ0YCAXrf9I0XFBI7b"],
      state: "connected",
      currentVTeam: "3268",
      participants: {
        "+19997770096": {
          id: "+19997770096",
          pType: "Customer",
          type: "Customer"
        },
        "aad323de-32d8-48c9-af6b-b68dfbabfe20": {
          name: "uui-agent2",
          pType: "Agent",
          teamName: "Team X",
          lastUpdated: 1597681728863,
          teamId: "962",
          joinTimestamp: 1597681728863,
          isConsulted: false,
          hasJoined: true,
          consultTimestamp: null,
          dn: "9997770194",
          queueId: "3268",
          id: "16839506-7c48-4a71-ba1b-d585e5d37607",
          sessionId: "319d8e83-2a75-4ef3-9cb8-1c845edb15a0",
          consultState: null,
          queueMgrId: "aqm",
          siteId: "473",
          type: "Agent",

```

```

        channelId: "ab8b3603-84a7-4f61-b987-9044b4d69cf2",
        wrapUpTimestamp: null,
        isWrapUp: false
    },
    interactionId: "49bcf26b-ec75-4351-89fa-55d54682c20c",
    orgId: "f111e3af-1a45-42ef-9erf-4562354b8a25",
    callAssociatedData: {
        customerName: {
            value: "test",
            name: "r",
            agentEditable: false,
            type: "s",
            displayName: "d"
        }
    },
    callAssociatedDetails: {
        virtualTeamName: "Queue - Telephony",
        ani: "+19997770096",
        ronaTimeout: "30",
        dn: "+12147659000",
        pathId: " StartCall PlayDone out out out",
        IvrPath: " EOI",
        subject: "",
        toAddress: "",
        inReplyTo: "",
        customerName: "",
        ccAddress: "",
        entryPointId: "",
        accountId: "",
        reasonCode: "",
        reason: "",
        references: "",
        contentType: "",
        date: "",
        replyToAddress: "",
        fromAddress: "",
        messageId: "",
        from: "",
        threadId: "",
        bccAddress: "",
        queueType: "",
        dnis: "",
        category: "",
        sourceNumber: "",
        sourcePage: "",
        appUser: "",
        customerNumber: ""
    },
    callProcessingDetails: {
        QMgrName: "aqm",
        pauseResumeEnabled: "true",
        taskToBeSelfServiced: "false",
        ani: "+19997770096",
        recordInProgress: "true",
        pauseDuration: "10",
        dnis: "+12147659000",
        tenantId: "133",
        QueueId: "3268",
        vteamId: "3268",
        jscripId: "AXCZ4c3mjkwgAuS7vSIU",
        customerName: "",
        virtualTeamName: "Queue - Telephony",
        ronaTimeout: "30",
    }
}

```

```

        category: "",
        reason: "",
        sourceNumber: "",
        sourcePage: "",
        appUser: "",
        customerNumber: "",
        reasonCode: "",
        IvrPath: " EOI",
        pathId: " StartCall PlayDone out out out",
        fromAddress: ""
    },
    media: {
        "49bcf26b-ec75-4351-89fa-55d54682c20c": {
            mediaResourceId: "49bcf26b-ec75-4351-89fa-55d54682c20c",
            mediaType: "telephony",
            mediaMgr: "vmm",
            participants: ["+19997770096", "16839506-7c48-4a71-balb-d585e5d37607"],

            mType: "mainCall",
            isHold: false,
            holdTimestamp: null
        }
    },
    owner: "16839506-7c48-4a71-balb-d585e5d37607",
    mediaChannel: "broadcloud",
    contactDirection: {
        type: "INBOUND"
    },
    callFlowParams: {
        Play2: {
            name: "Play2",
            qualifier: "",
            description: "(A valid text.)",
            valueDataType: "string",
            value: "Welcome to Agent Team Space"
        },
        Queue3: {
            name: "Queue3",
            qualifier: "vteam",
            description: "(vteam, A valid VTeam.)",
            valueDataType: "string",
            value: "3268"
        }
    }
},
interactionId: "49bcf26b-ec75-4351-89fa-55d54682c20c",
orgId: "f111e3af-1a45-42ef-9erf-4562354b8a25",
queueMgr: "aqm",
type: "AgentContactAssigned",
destAgentId: "16839506-7c48-4a71-balb-d585e5d37607",
consultMediaResourceId: "",
owner: "",
isConferencing: false
    };
})
// => Add custom task object in Desktop store

```

getTaskMap()

Retrieves the full list of tasks that are assigned to an agent at a given time.

Example

```
const currentTaskMap = await Desktop.actions.getTaskMap()
// => Get current task map from Desktop store
```

Returns

{Map} A list of results.

Example Response

```
Desktop Store TaskMap:
  Map(1) {
    "851a8e81-6150-11eb-9a47-f1ca3756d4bd" => {
      ...}, "eed4b53a-614f-11eb-9a47-8be6439707ee" => {
      ...}
    }
  [
    [Entries]
  ] {
    "851a8e81-6150-11eb-9a47-f1ca3756d4bd" => Object
  }
key: "851a8e81-6150-11eb-9a47-f1ca3756d4bd"
value:
  agentId: "7c867aa9-ec768-341a-b767-e5hd6ae7g701"
  eventType: "RoutingMessage"
  interaction: {
    callAssociatedData: {
      ...},
    callAssociatedDetails: {
      ...},
    callFlowParams: {
      ...},
    callProcessingDetails: {
      ...},
    contactDirection: {
      ...},
    ...
  }
  interactionId: "851a8e81-6150-11eb-9a47-f1ca3756d4bd"
  mediaResourceId: "4D44FF32-A585-4DD8-8422-4920FC97066A@cisco.com"
  orgId: "f111e3af-1a45-42ef-9erf-4562354b8a25"
  queueMgr: "aqm"
  trackingId: "69773a10-6163-11eb-9fb9-8dadf7aee433"
  type: "AgentContactAssigned"
```



CHAPTER 7

Logger Module

The `Desktop.logger` module creates and maintains the client-side log messages for third-party widgets.

With individually set up logger, you can integrate with the Agent Desktop logging system (that is being used for telemetry and problem report) and at the same time maintain a clear origin definition. We recommend that you create a logger instance for your widgets only once per project. Create a logger instance in a `utils.ts` file and reuse throughout the components.

Example

```
// utils.ts file
import {
  Desktop
} from "@wxc-cc-desktop/sdk";

export const logger = Desktop.logger.createLogger("my-custom-component");

// Component.ts file
import {
  logger
} from "../utils.ts";

logger.info("Info test"); // logger.info => 2020-12-16 13:11:04:971["my-custom-component",
  "Info test"]
logger.warn("Warn test"); // logger.info => 2020-12-16 13:11:04:971["my-custom-component",
  "Warn test"]
logger.error("Error test"); // logger.info => 2020-12-16 13:11:04:971["my-custom-component",
  "Error test"]
```

For more information on the widget starter example, see [Cisco Webex Contact Center Widget Starter](#).

You can obtain logs in a JSON format or as a downloadable *.log file specifically for your widget. The following are the available options to download the logs:

```
// Download logs as a JSON file for "my-custom-component" prefix:
Desktop.logger.browserDownloadLogsJson("my-custom-component");

// Download logs as a Text file for "my-custom-component" prefix:
Desktop.logger.browserDownloadLogsText("my-custom-component");

// Get logs as Objects collection for "my-custom-component" prefix:
Desktop.logger.getLogsCollection("my-custom-component");

// Get logs as base64 encoded url ready to put into link href to initiate browser download
as a JSON file for "my-custom-component" prefix:
Desktop.logger.getLogsJsonUrl("my-custom-component");

// Get logs as base64 encoded url ready to put into link href to initiate browser download
```

```
as a Text file for "my-custom-component" prefix:
Desktop.logger.getLogsTextUrl("my-custom-component");

// Cleanup logs from Local Storage for "my-custom-component" prefix:
Desktop.logger.cleanupPrefixedLogs("my-custom-component");
```

You can also download logs from the Agent Desktop. For more information, see the *Download Error Report* section in the *Working with Agent Desktop* chapter of the [Cisco Webex Contact Center Agent Desktop User Guide](#).

- [Methods, on page 40](#)

Methods

createLogger(my-custom-component)

Initializes the logger for third-party widgets that help in logging related information to a particular component. Different methods are used based on the log type. The messages such as info, warn, and error are logged in the browser console.

Example

```
const loggerOne = Desktop.logger.createLogger("my-custom-component-one");
const loggerTwo = Desktop.logger.createLogger("my-custom-component-two");

loggerOne.info("Info test"); // console.log => "my-custom-component: Info:test"
loggerTwo.warn("Warn test"); // console.log => "my-custom-component: Warn:test"
loggerOne.error("Error test"); // console.log => "my-custom-component: Error:test"
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes

browserDownloadLogsJson(my-custom-component)

Downloads logs as a JSON file for the my-custom-component prefix.

Example

```
Desktop.logger.browserDownloadLogsJson("my-custom-component");
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes

browserDownloadLogsText(my-custom-component)

Downloads logs as a text file for the my-custom-component prefix.

Example

```
Desktop.logger.browserDownloadLogsText("my-custom-component");
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes

getLogsCollection(my-custom-component)

Retrieves logs as a collection of objects for the `my-custom-component` prefix.

Example

```
Desktop.logger.getLogsCollection("my-custom-component");
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes

getLogsJsonUrl(my-custom-component)

Retrieves logs as a Base64 encoded URL to initiate browser download as a JSON file for the `my-custom-component` prefix.

Example

```
Desktop.logger.getLogsJsonUrl("my-custom-component");
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes

getLogsTextUrl(my-custom-component)

Retrieves logs as a Base64 encoded URL to initiate browser download as a text file for the `my-custom-component` prefix.

Example

```
Desktop.logger.getLogsTextUrl("my-custom-component");
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes

cleanupPrefixedLogs(my-custom-component)

Cleans up logs from the local storage for the `my-custom-component` prefix.

Example

```
Desktop.logger.cleanupPrefixedLogs("my-custom-component");
```

Parameters

Name	Type	Description	Required
my-custom-component	String	Name of the component. Example: email component, chat component.	Yes



CHAPTER 8

Agent State Information Module

The `Desktop.agentStateInfo` module listens for the latest data updates of agent related information.

Example

```
import {
    Service
} from "@wxcc-desktop/sdk-types";

type LatestInfoData = {
    teamId ? : string;
    teamName ? : string;
    dn ? : string;
    status ? : string;
    subStatus ? : string;
    idleCodes ? : Service.Aqm.Configs.Entity[];
    wrapupCodes ? : Service.Aqm.Configs.Entity[];
    outDialRegex ? : string;
    isOutboundEnabledForTenant ? : boolean;
    isOutboundEnabledForAgent ? : boolean;
};
```

The following table lists the payload details:

Name	Type	Description	Required
teamId	String	Unique identifier of the team.	Yes
teamName	String	Name of the team.	Yes
dn	String	Dial number (DN) to the entry point.	Yes
status	String	Agent log in status such as LoggedIn.	Yes
subStatus	String	Agent availability status such as Available.	Yes
idleCodes	String	The idle reason codes indicating that the agent is not ready to accept any routed requests. Example: Idle, Coffee break, Meeting, Tea	Yes
wrapupCodes	String	The wrap up reason codes indicating that the agent has ended the interactions with the customer.	Yes
outDialRegex	String	Outdial regular expression validity.	Yes

Name	Type	Description	Required
isOutboundEnabledForTenant	Boolean	Determines whether the outbound feature is enabled for the tenant. <ul style="list-style-type: none"> • True—Enables the outbound feature for the tenant. • False—Disables the outbound feature for the tenant. 	Yes
isOutboundEnabledForAgent	Boolean	Determines whether the outbound feature is enabled for the agent. <ul style="list-style-type: none"> • True—Enables the outbound feature for the agent. • False—Disables the outbound feature for the agent. 	Yes

- [Methods, on page 44](#)
- [Events, on page 46](#)

Methods

latestData

Fetches the latest user related information.

Example

```
const latestData: LatestInfoData = Desktop.agentStateInfo.latestData;
```

Returns

{Object} Lists the latest user related information.

Example Response

```
{
  "agentName": "John Doe",
  "teamName": "Sales",
  "teamId": "964",
  "dn": "8895579172",
  "status": "LoggedIn",
  "subStatus": "Idle",
  "idleCodes": [{
    "id": "1646",
    "isDefault": false,
    "isSystem": true,
    "name": "Aux on Login"
  }],
  "wrapupCodes": [{
    "id": "2065",
    "isDefault": false,
    "isSystem": false,
  }],
}
```

```

        "name": "Wrapping up as customer disconnected"
    }],
    "outDialRegex": "([0-9a-zA-Z]+[-._])*[0-9a-zA-Z]+",
    "isOutboundEnabledForTenant": true,
    "isOutboundEnabledForAgent": true
}

```

stateChange(state, auxCodeIdArray)

Retrieves the state change of the user.

Example

```
const state = await Desktop.agentStateInfo.stateChange({ state: s, auxCodeIdArray: "0" });
```

Parameters

Name	Type	Description	Required
state	String	Current agent state.	Yes
auxCodeIdArray	String	Unique identifier of the agent state.	Yes

Returns

{Object} Lists the agent state change information.

Example Response

```

{
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "agentSessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
    "auxCodeId": "0",
    "connectedChannels": [
      "38136791-cf79-4663-a5d0-39b0d094abf8"
    ],
    "eventType": "AgentDesktopMessage",
    "lastIdleCodeChangeTimestamp": null,
    "lastStateChangeReason": "",
    "lastStateChangeTimestamp": 1612245463183,
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "status": "LoggedIn",
    "subStatus": "Available",
    "trackingId": "9008a4a0-651b-11eb-a97d-0d76c582d799",
    "type": "AgentStateChangeSuccess"
  },
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "trackingId": "notifs_aa848b62-4fcf-4d4c-95ad-1233bb7f899c",
  "type": "AgentStateChange"
}

```

fetchAddressBooks()

Retrieves the address book details.

Example

```
const books = await Desktop.agentStateInfo.fetchAddressBooks()
```

Returns

{Array} The array of objects.

Example Response

```
type AddressBooks = {
  speedDials: [{
    desc: 'Jane Doe',
    dn: '9997770094'
  },
  {
    desc: 'John Doe',
    dn: '9997770095'
  },
]
};
type Address = {
  desc: string;
  dn: string;
  phoneBookName ? : string;
};
```

Events

addEventListener

Listens to an event named `updated` that is logged if the `dn`, `status`, or `subStatus` field is changed.

For more information, see [addEventListener\(event, handler\)](#).

Example

```
Desktop.agentStateInfo.addEventListener("updated", updatedList =>
  console.log(updatedList) [{
    "name": "dn",
    "value": "+12580258011",
    "oldValue": ""
  }, {
    "name": "status",
    "value": "LoggedIn",
    "oldValue": "DefaultState"
  }, {
    "name": "subStatus",
    "value": "Available",
    "oldValue": ""
  }] *
  /
)
```



CHAPTER 9

Agent Contact Module

The `Desktop.agentContact` module makes requests and listens to notification events related to the agent-contact entity, such as the arrival of a new task.

Example

```
import {
    Desktop
} from "@wxcc-desktop/sdk";

//...

/*
    Supposing Desktop.config.init({...}) was called
*/

// List of available agent-contact aqm reqs:
await Desktop.agentContact.accept({
    ...
});
await Desktop.agentContact.consultAccept({
    ...
});
await Desktop.agentContact.buddyAgents({
    ...
});
await Desktop.agentContact.end({
    ...
});
await Desktop.agentContact.consultEnd({
    ...
});
await Desktop.agentContact.cancelCtq({
    ...
});
await Desktop.agentContact.wrapup({
    ...
});
await Desktop.agentContact.vteamTransfer({
    ...
});
await Desktop.agentContact.blindTransfer({
    ...
});
await Desktop.agentContact.hold({
    ...
});
await Desktop.agentContact.unHold({
```

```

    ...
  });
  await Desktop.agentContact.consult({
    ...
  });
  await Desktop.agentContact.decline({
    ...
  });
  await Desktop.agentContact.consultTransfer({
    ...
  });
  await Desktop.agentContact.vteamList({
    ...
  });
  await Desktop.agentContact.pauseRecording({
    ...
  });
  await Desktop.agentContact.resumeRecording({
    ...
  });

  // List of available agent-contact aqm notifs events:
  Desktop.agentContact.addEventListener("eAgentContact", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentContactAssigned", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentContactAssignFailed", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentContactEnded", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentContactWrappedUp", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentOfferContact", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentOfferContactRona", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentOfferConsult", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentWrapup", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentContactHeld", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentContactUnHeld", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eCallRecordingStarted", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eResumeRecording", msg => console.log(msg));
  Desktop.agentContact.addEventListener("ePauseRecording", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eConsultTransfer", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentblindTransferred", msg => console.log(msg));

  Desktop.agentContact.addEventListener("eAgentvteamTransfer", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultCreated", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultConferenced", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultEnded", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentCtqCancelled", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultConferenceEnded", msg =>
    console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsulting", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultFailed", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultEndFailed", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentCtqFailed", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentCtqCancelFailed", msg => console.log(msg));
  Desktop.agentContact.addEventListener("eAgentConsultConferenceEndFailed", msg =>
    console.log(msg));

  // Module supports removing added listeners like:
  const listener = msg => console.log(msg);
  Desktop.agentContact.addEventListener("eAgentContact", listener);
  Desktop.agentContact.removeEventListener("eAgentContact", listener);

  // Module supports one-time added listeners like:
  Desktop.agentContact.addOnceEventListener("eAgentContact", listener);
  Desktop.agentContact.removeOnceEventListener("eAgentContact", listener);

  // Module supports removing all listeners like:

```



```
Desktop.agentContact.removeAllEventListeners();
```

Example Response

```
// Generic Response Example form of data is :
data: {
  mediaResourceId: string;
  eventType: string;
  agentId: string;
  destAgentId: string;
  trackingId: string;
  consultMediaResourceId: string;
  interaction: {
    isFcManaged: boolean;
    isTerminated: boolean;
    mediaType: "email" | "chat" | "telephony" | "social" | "sms" | "facebook" | string;

    previousVTeams: string[];
    state: string;
    currentVTeam: string;
    participants: any; // todo
    interactionId: string;
    orgId: string;
  },
  interactionId: string;
  orgId: string;
  owner: string;
  queueMgr: string;
  type: string;
  ronaTimeout ? : number;
  isConsulted ? : boolean;
  isConferencing: boolean;
  updatedBy ? : string;
  destinationType ? : string;
}
```

- [Methods, on page 49](#)
- [Asynchronous Events, on page 104](#)

Methods

accept()

Accepts an incoming task.

Example

```
await Desktop.agentContact.accept({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
});
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes

Returns

{Object} The value that corresponds to the task.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentContactAssigned`; else, `AgentContactAssignFailed`.

Example Response

```

const acceptResponse = {
  "data": {
    "agentId": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "value": "janedoe@gmail.com"
        },
        "category": {
          "value": "Sales"
        },
        "customerName": {
          "value": "Jane Doe"
        },
        "dn": {
          "value": "Desktop"
        },
        "entryPointId": {
          "value": "AXCqFyz1G2pKap9PI-mW"
        },
        "guestId": {
          "value":
            "Y21zY29zcGFyazovL3VzL1BFTE1BMRS83ZjA3YjJmMC1jNTMyLTQ5MDktYTktNi0yYTA3MTVmZGIwMTA"
        },
        "mediaChannel": {
          "value": "web"
        },
        "reason": {
          "value": "Test"
        },
        "reasonCode": {
          "value": "Sales"
        },
        "ronaTimeout": {
          "value": "30"
        },
        "roomTitle": {
          "value": "Help Jane Doe with Sales"
        },
        "taskToBeSelfServiced": {
          "value": "false"
        },
        "templateId": {
          "value": "b57990c0-5ec6-11ea-96ee-5df5aef56329"
        },
        "templateName": {
          "value": "Desktop"
        },
        "virtualTeamName": {
          "value": "Chat_Queue"
        }
      }
    }
  }
}

```

```

    },
    "callAssociatedDetails": {
      "ani": "janedoe@gmail.com",
      "category": "Sales",
      "customerName": "Jane Doe",
      "dn": "Desktop",
      "entryPointId": "AXCqFyz1G2pKap9PI-mW",
      "guestId":
"Y21zY29zcGFyazovL3VzL1BFT1BMRS83ZjA3YjJmMC1jNTMyLTQ5MDktYTktZni0yYTA3MTVmZGIwMTA",
      "mediaChannel": "web",
      "reason": "Test",
      "reasonCode": "Sales",
      "ronaTimeout": "30",
      "roomTitle": "Help Jane Doe with Sales",
      "taskToBeSelfServiced": "false",
      "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
      "templateName": "Desktop",
      "virtualTeamName": "Chat_Queue"
    },
    "callFlowParams": {
      "Automation": {
        "description": "(vteam, A valid VTeam.)",
        "name": "Automation",
        "qualifier": "vteam",
        "value": "3270",
        "valueDataType": "string"
      },
      "Debit": {
        "description": "(vteam, A valid VTeam.)",
        "name": "Debit",
        "qualifier": "vteam",
        "value": "3270",
        "valueDataType": "string"
      },
      "Sales": {
        "description": "(vteam, A valid VTeam.)",
        "name": "Sales",
        "qualifier": "vteam",
        "value": "3270",
        "valueDataType": "string"
      }
    },
    "callProcessingDetails": {
      "QMgrName": "aqm",
      "QueueId": "3270",
      "ani": "janedoe@gmail.com",
      "category": "Sales",
      "customerName": "Jane Doe",
      "dnis": "Desktop",
      "entryPointId": "AXCqFyz1G2pKap9PI-mW",
      "guestId":
"Y21zY29zcGFyazovL3VzL1BFT1BMRS83ZjA3YjJmMC1jNTMyLTQ5MDktYTktZni0yYTA3MTVmZGIwMTA",
      "mediaChannel": "web",
      "reason": "Test",
      "reasonCode": "Sales",
      "ronaTimeout": "30",
      "roomTitle": "Help Jane Doe with Sales",
      "taskToBeSelfServiced": "false",
      "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
      "templateName": "Desktop",
      "tenantId": "133",
      "virtualTeamName": "Chat_Queue",
      "vteamId": "AXCqFyz1G2pKap9PI-mW"
    },
  },

```

accept()

```

        "contactDirection": {
            "type": "INBOUND"
        },
        "currentVTeam": "3270",
        "interactionId": "59dbbca6-4194-11eb-881c-253923a967e0",
        "isFcManaged": false,
        "isTerminated": false,
        "media": {
            "Y21zY29zcGFyazovL3VzL1JPT00vNjVjOWVlYjAtNDE5NC0xMWViLWE1N2QtZWY2N2MwZTM5YmFh": {
                "holdTimestamp": null,
                "isHold": false,
                "mType": "mainCall",
                "mediaMgr": "cmm",
                "mediaResourceId":
            "Y21zY29zcGFyazovL3VzL1JPT00vNjVjOWVlYjAtNDE5NC0xMWViLWE1N2QtZWY2N2MwZTM5YmFh",
                "mediaType": "chat",
                "participants": ["janedoe@gmail.com",
            "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed"]
                }
        },
        "mediaChannel": "web",
        "mediaType": "chat",
        "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
        "outboundType": null,
        "owner": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
        "participants": {
            "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed": {
                "channelId": "bff9350a-9d2a-489a-a8d5-d9dacea4b692",
                "consultState": null,
                "consultTimestamp": null,
                "dn": "9997770110",
                "hasJoined": true,
                "id": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
                "isConsulted": false,
                "isWrapUp": false,
                "joinTimestamp": 1608339210824,
                "lastUpdated": 1608339210824,
                "name": "uuiip-agent4 uuiip-agent4",
                "pType": "Agent",
                "queueId": "3270",
                "queueMgrId": "aqm",
                "sessionId": "102d2096-bcc0-45bb-a583-a02f6c188071",
                "siteId": "473",
                "teamId": "1940",
                "teamName": "Medical Help",
                "type": "Agent",
                "wrapUpTimestamp": null
            },
            "janedoe@gmail.com": {
                "id": "janedoe@gmail.com",
                "pType": "Customer",
                "type": "Customer"
            }
        },
        "previousVTeams": [],
        "state": "connected"
    },
    "interactionId": "59dbbca6-4194-11eb-881c-253923a967e0",
    "mediaResourceId":
    "Y21zY29zcGFyazovL3VzL1JPT00vNjVjOWVlYjAtNDE5NC0xMWViLWE1N2QtZWY2N2MwZTM5YmFh",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "queueMgr": "aqm",
    "trackingId": "9c9b97e0-4194-11eb-b819-7b8dd50ee0cd",

```

```

    "type": "AgentContactAssigned"
  },
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "trackingId": "notifs_02db68c8-a5db-4028-95bb-1aebd6f87609",
  "type": "RoutingMessage"
}

```

consultAccept()

Accepts a consult task.

Example

```

await Desktop.agentContact.consultAccept({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
});

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes

Returns

{Object} The value that corresponds to the task.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentConsulting`; else, `AgentContactAssignFailed`.

Example Response

```

const consultAcceptResponse = {
  "data": {
    "agentId": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
    "consultMediaResourceId": "c1b507d0-9aac-4500-a9f8-50f30eba4310",
    "destAgentId": "ff7dbe19-fa3f-4c63-8a1c-04dcffef8e4f",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "": {
          "agentEditable": false,
          "displayName": "",
          "name": "",
          "type": "STRING",
          "value": ""
        },
        "IvrPath": {
          "agentEditable": false,
          "displayName": "IvrPath",
          "name": "IvrPath",
          "type": "STRING",
          "value": " EOI"
        },
        "ani": {
          "agentEditable": false,
          "displayName": "ani",

```

```

        "name": "ani",
        "type": "STRING",
        "value": "*****"
    },
    "dn": {
        "agentEditable": false,
        "displayName": "dn",
        "name": "dn",
        "type": "STRING",
        "value": "*****"
    },
    "nextVteamId": {
        "agentEditable": false,
        "displayName": "nextVteamId",
        "name": "nextVteamId",
        "type": "STRING",
        "value": "Queue"
    },
    "pathId": {
        "agentEditable": false,
        "displayName": "pathId",
        "name": "pathId",
        "type": "STRING",
        "value": " StartCall PlayDone"
    },
    "ronaTimeout": {
        "agentEditable": false,
        "displayName": "ronaTimeout",
        "name": "ronaTimeout",
        "type": "STRING",
        "value": "30"
    },
    "virtualTeamName": {
        "agentEditable": false,
        "displayName": "virtualTeamName",
        "name": "virtualTeamName",
        "type": "STRING",
        "value": "Queue-1"
    }
},
"callAssociatedDetails": {
    "": "",
    "IvrPath": "EOI",
    "ani": "*****",
    "dn": "*****",
    "nextVteamId": "Queue",
    "pathId": " StartCall PlayDone",
    "ronaTimeout": "30",
    "virtualTeamName": "Queue-1"
},
"callFlowParams": {
    "Queue": {
        "description": "(vteam, A valid VTeam.)",
        "name": "Queue",
        "qualifier": "vteam",
        "value": "1336",
        "valueDataType": "string"
    },
    "WelcomePrompt": {
        "description": "(mediaFile, A valid media file.)",
        "name": "WelcomePrompt",
        "qualifier": "mediaFile",
        "value": "GAwelcome.wav",
        "valueDataType": "string"
    }
}

```

```

    }
  },
  "callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "1336",
    "ani": "*****",
    "dnis": "*****",
    "isPaused": "false",
    "jscriptId": "AW7HP3QNB44q0sL-cTVz",
    "pauseDuration": "30",
    "pauseResumeEnabled": "true",
    "recordInProgress": "true",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "82",
    "virtualTeamName": "Queue-1",
    "vteamId": "1336"
  },
  "contactDirection": {
    "type": "INBOUND"
  },
  "currentVTeam": "1336",
  "interactionId": "1277da79-dc49-439c-ab72-f00453a1ff71",
  "isFcManaged": false,
  "isTerminated": false,
  "media": {
    "1277da79-dc49-439c-ab72-f00453a1ff71": {
      "holdTimestamp": 1612429578817,
      "isHold": true,
      "mType": "mainCall",
      "mediaMgr": "vmm",
      "mediaResourceId": "1277da79-dc49-439c-ab72-f00453a1ff71",
      "mediaType": "telephony",
      "participants": [
        "*****",
        "9b036d89-930c-4187-a5bd-5dbb1439fe41"
      ]
    },
    "c1b507d0-9aac-4500-a9f8-50f30eba4310": {
      "holdTimestamp": null,
      "isHold": false,
      "mType": "consult",
      "mediaMgr": "vmm",
      "mediaResourceId": "c1b507d0-9aac-4500-a9f8-50f30eba4310",
      "mediaType": "telephony",
      "participants": [
        "ff7dbel9-fa3f-4c63-8a1c-04dcffef8e4f",
        "9b036d89-930c-4187-a5bd-5dbb1439fe41",
        "ff7dbel9-fa3f-4c63-8a1c-04dcffef8e4f"
      ]
    }
  },
  "mediaChannel": "broadcloud",
  "mediaType": "telephony",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "outboundType": null,
  "owner": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
  "participants": {
    "*****": {
      "id": "*****",
      "pType": "Customer",
      "type": "Customer"
    },
    "9b036d89-930c-4187-a5bd-5dbb1439fe41": {

```

consultAccept()

```

        "channelId": "3de6706f-95b7-485a-9304-30928f7ee52e",
        "consultState": "consulting",
        "consultTimestamp": 1612429579220,
        "dn": "*****",
        "hasJoined": true,
        "id": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
        "isConsulted": true,
        "isWrapUp": false,
        "joinTimestamp": 1612429549032,
        "lastUpdated": 1612429549032,
        "name": "user1",
        "pType": "Agent",
        "queueId": "1336",
        "queueMgrId": "aqm",
        "sessionId": "90896c91-0f77-4bc6-aec3-488a34ca88d3",
        "siteId": "205",
        "teamId": "598",
        "teamName": "integ-test-team",
        "type": "Agent",
        "wrapUpTimestamp": null
    },
    "ff7dbe19-fa3f-4c63-8a1c-04dcffef8e4f": {
        "channelId": "fb704337-71c3-4802-a03b-84a2d18eeaa0",
        "consultState": "consulting",
        "consultTimestamp": 1612429579220,
        "dn": "*****",
        "hasJoined": true,
        "id": "ff7dbe19-fa3f-4c63-8a1c-04dcffef8e4f",
        "isConsulted": true,
        "isWrapUp": false,
        "joinTimestamp": 1612429579200,
        "lastUpdated": 1612429579200,
        "name": "user2",
        "pType": "Agent",
        "queueId": "1336",
        "queueMgrId": "aqm",
        "sessionId": "57323f8b-9460-45fa-aaf6-4ca51c327af8",
        "siteId": "205",
        "teamId": "598",
        "teamName": "integ-test-team",
        "type": "Agent",
        "wrapUpTimestamp": null
    }
},
"previousVTeams": [
    "AW6hlAVRrZSGnl185x28"
],
"state": "consulting",
"workflowManager": null
},
"interactionId": "1277da79-dc49-439c-ab72-f00453a1ff71",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"trackingId": "e7c032be-b18e-4b63-8ab2-6baf50b137b1",
"type": "AgentConsulting"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_3464be63-83e7-4b2d-94d5-a63fddc108d5",
"type": "RoutingMessage"
}

```


end()

Ends an interaction for a specific task.

Example

```
await Desktop.agentContact.end({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
});
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes

Returns

{Object} The value that corresponds to the task.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentWrapup` or `ContactEnded`; else, `ContactEndFailed`.

Example Response

```
const endResponse = {
  "data": {
    "agentsPendingWrapUp": ["7d12d9ea-e8e0-41ee-81bf-c11a685b64ed"],
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "value": "janedoe@gmail.com"
        },
        "category": {
          "value": "Sales"
        },
        "customerName": {
          "value": "Jane Doe"
        },
        "dn": {
          "value": "Desktop"
        },
        "entryPointId": {
          "value": "AXCqFyz1G2pKap9PI-mW"
        },
        "guestId": {
          "value":
            "Y21zY29zcGFyazovL3VzL1BFT1BMRS81NzNiODg1Mi1kZjM3LTRlODEtOGMxM1mZjkyMTE3MmQ1NzY"
        },
        "mediaChannel": {
          "value": "web"
        },
        "reason": {
          "value": "Test"
        },
        "reasonCode": {
          "value": "Sales"
        }
      }
    }
  }
};
```

end()

```

        "ronaTimeout": {
            "value": "30"
        },
        "roomTitle": {
            "value": "Help Jane Doe with Sales"
        },
        "taskToBeSelfServiced": {
            "value": "false"
        },
        "templateId": {
            "value": "b57990c0-5ec6-11ea-96ee-5df5aef56329"
        },
        "templateName": {
            "value": "Desktop"
        },
        "virtualTeamName": {
            "value": "Chat_Queue"
        }
    },
    "callAssociatedDetails": {
        "ani": "janedoe@gmail.com",
        "category": "Sales",
        "customerName": "Jane Doe",
        "dn": "Desktop",
        "entryPointId": "AXCqFyz1G2pKap9PI-mW",
        "guestId":
        "Y2lzY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1M1kZjM3LTRlODEtOGMxM1mZjkyMTE3MmQ1NzY",
        "mediaChannel": "web",
        "reason": "Please help",
        "reasonCode": "Sales",
        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Sales",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "virtualTeamName": "Chat_Queue"
    },
    "callFlowParams": {
        "Automation": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Automation",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Debit": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Debit",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Sales": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Sales",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        }
    },
    "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3270",
        "ani": "janedoe@gmail.com",

```

```

        "category": "Sales",
        "customerName": "Jane Doe",
        "dnis": "Desktop",
        "entryPointId": "AXCqFyz1G2pKap9PI-mW",
        "guestId":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1M1lkZjM3LTRlODEtOGMxM1lmZjkyMTE3MmQ1NzY",
        "mediaChannel": "web",
        "pauseDuration": "10",
        "pauseResumeEnabled": "true",
        "reason": "Please help",
        "reasonCode": "Sales",
        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Sales",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "tenantId": "133",
        "virtualTeamName": "Chat_Queue",
        "vteamId": "AXCqFyz1G2pKap9PI-mW"
    },
    "contactDirection": {
        "type": "INBOUND"
    },
    "currentVTeam": "3270",
    "interactionId": "a66d05ab-40ab-11eb-b59a-7d3326b3ffcl",
    "isFcManaged": false,
    "isTerminated": true,
    "media": {},
    "mediaChannel": "web",
    "mediaType": "chat",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": null,
    "owner": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "participants": {
        "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed": {
            "channelId": "ff0a5296-8aa2-483b-8c50-577fedb1bfde",
            "consultState": null,
            "consultTimestamp": null,
            "dn": "9997770110",
            "hasJoined": true,
            "id": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
            "isConsulted": false,
            "isWrapUp": false,
            "joinTimestamp": 1608239195992,
            "lastUpdated": 1608239195992,
            "name": "uui-agent4 uui-agent4",
            "pType": "Agent",
            "queueId": "3270",
            "queueMgrId": "aqm",
            "sessionId": "b8c096ab-f9a0-49be-9efd-f8c78e1a7061",
            "siteId": "473",
            "teamId": "962",
            "teamName": "Team X",
            "type": "Agent",
            "wrapUpTimestamp": null
        },
        "janedoe@gmail.com": {
            "id": "janedoe@gmail.com",
            "pType": "Customer",
            "type": "Customer"
        }
    },
    "previousVTeams": [],
    "state": "connected"

```

```

    },
    "interactionId": "a66d05ab-40ab-11eb-b59a-7d3326b3ffc1",
    "mediaResourceId":
    "Y2lzY29zcGFyYzovL3VzL1JPT00vYWZNTA5YzAtNDBhYi0xMWViLWE1N2QtZWY2N2MwZTMzMmYmFh",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "queueMgr": "aqm",
    "trackingId": "4eb68870-40ac-11eb-8866-d7ef6ff89aa6",
    "type": "ContactEnded"
  },
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "trackingId": "notifs_3cda2c34-1410-4476-9c5c-5d779a615b8a",
  "type": "RoutingMessage"
}

```

buddyAgents()

Fetches the list of agents available for consult call and conference call.

Example

```

await Desktop.agentContact.buddyAgents({
  data: {
    agentProfileId: "AXCLfZhH9SloTdqE1OFw",
    channelName: "chat",
    state: "Available"
  }
});

```

The following table lists the payload details:

Name	Type	Description	Required
data	Object	Options of the buddy agents.	Yes
-->agentProfileId	String	Unique identifier of the agent profile.	Yes
-->channelName	String	The media channel type such as chat.	Yes
-->state	String	The agent availability status.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `BuddyAgents`; else, `BuddyAgentsRetrieveFailed`.

Example Response

```

type BuddyAgentResponse = {
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "agentList": [],
    "agentSessionId": "25f31485-f41c-4cca-abe5-e12b80735715",
    "eventType": "AgentDesktopMessage",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "trackingId": "1a2737c0-6704-11eb-9a89-971a3ca976b4",

```

```

        "type": "BuddyAgents"
    },
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "trackingId": "notifs_49119bed-6dbc-48e6-89fa-9dde8da3fc3e",
    "type": "BuddyAgents"
}

```

consult()

Initiates a consult request with one of the agents from the list of buddyAgents.

Example

```

await Desktop.agentContact.consult({
  interactionId: "a26d035d - b547 - 11 ea - 984 b - d3128111ce6b",
  data: {
    agentId: "43 bfcdef - 5551 - 45 b1 - a8f9 - a0b33e66e3fe",
    destAgentId: "f795f41f - 3782 - 44 fa - 97 a4 - c8b4dc029477",
    mediaType: "chat"
  },
  url: "consult"
});
// consult Payload Type
type ConsultPayload = {
  interactionId: string;
  data: ConsultPayload | ConsultDN | ConsultAgent | ConsultQueue;
  url: string;
}

type ConsultPayload = {
  agentId: string;
  destAgentId: string | undefined;
  mediaType ? : string | undefined;
}

type ConsultDN = {
  destAgentId: string;
  destinationType: string;
  mediaType: string;
  trackingId ? : string;
};

type ConsultAgent = {
  agentId: string;
  destAgentId: string;
  destAgentDN ? : string;
  destAgentTeamId ? : string;
  destSiteId ? : string;
  mediaType: string;
  trackingId: string;
};

type ConsultQueue = {
  agentId: string;
  queueId: string;
  trackingId: string;
};

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes

Name	Type	Description	Required
data	Object	Options of the consult request.	Yes
-->agentId	String	Unique identifier of the agent.	Yes
-->destAgentId	String	Unique identifier of the consult request destination agent.	Yes
-->mediaType	String	The media channel type such as chat.	Yes
-->destinationType	String	The type of consult request to be placed. The destination type can be Agent, Queue, or DN.	Yes
-->trackingId	String	Unique identifier based on the consult request type.	Yes
-->destAgentDN	String	The dial number of the destination agent.	Yes
-->destAgentTeamId	String	Unique identifier of the team to which the destination agent belongs.	Yes
-->destSiteId	String	Unique identifier of the destination site.	Yes
-->queueId	String	Unique identifier of the queue.	Yes
url	String	The URL of the consult. Example, consult, ctq.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentConsultCreated`; else, `AgentCtqFailed`, or `AgentConsultFailed`.

Example Response

```
const consultResponse = {
  "data": {
    "agentId": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "consultMediaResourceId":
"consult_Y21zY29zcGFyazovL3VzL1JPT00vNjJkYjBmMzAtNDZhZi0xMWVlLWE3ZGItYjkyYTZkMmI5NTI1",
    "destAgentId": "aad323de-32d8-48c9-af6b-b68dfbabfe20",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "value": "janedoe@gmail.com"
        },
        "category": {
          "value": "Sales"
        },
        "customerName": {
          "value": "Jane Doe"
        }
      }
    }
  }
}
```

```

    },
    "dn": {
      "value": "Desktop"
    },
    "entryPointId": {
      "value": "AXCqFyz1G2pKap9PI-mW"
    },
    "guestId": {
      "value":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1M1lkZjM3LTRlODEtOGMxM1mZjkyMTE3MmQ1NzY"
    },
    "mediaChannel": {
      "value": "web"
    },
    "reason": {
      "value": "test"
    },
    "reasonCode": {
      "value": "Sales"
    },
    "ronaTimeout": {
      "value": "30"
    },
    "roomTitle": {
      "value": "Help Jane Doe with Sales"
    },
    "taskToBeSelfServiced": {
      "value": "false"
    },
    "templateId": {
      "value": "b57990c0-5ec6-11ea-96ee-5df5aef56329"
    },
    "templateName": {
      "value": "Desktop"
    },
    "virtualTeamName": {
      "agentEditable": false,
      "displayName": "virtualTeamName",
      "name": "virtualTeamName",
      "type": "STRING",
      "value": "Chat_Queue"
    }
  },
  "callAssociatedDetails": {
    "ani": "janedoe@gmail.com",
    "category": "Sales",
    "customerName": "Jane Doe",
    "dn": "Desktop",
    "entryPointId": "AXCqFyz1G2pKap9PI-mW",
    "guestId":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1M1lkZjM3LTRlODEtOGMxM1mZjkyMTE3MmQ1NzY",
    "mediaChannel": "web",
    "reason": "test",
    "reasonCode": "Sales",
    "ronaTimeout": "30",
    "roomTitle": "Help Jane Doe with Sales",
    "taskToBeSelfServiced": "false",
    "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
    "templateName": "Desktop",
    "virtualTeamName": "Chat_Queue"
  },
  "callFlowParams": {
    "Automation": {
      "description": "(vteam, A valid VTeam.)",

```

```

        "name": "Automation",
        "qualifier": "vteam",
        "value": "3270",
        "valueDataType": "string"
    },
    "Debit": {
        "description": "(vteam, A valid VTeam.)",
        "name": "Debit",
        "qualifier": "vteam",
        "value": "3270",
        "valueDataType": "string"
    },
    "Sales": {
        "description": "(vteam, A valid VTeam.)",
        "name": "Sales",
        "qualifier": "vteam",
        "value": "3270",
        "valueDataType": "string"
    }
},
"callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "3270",
    "ani": "janedoe@gmail.com",
    "category": "Sales",
    "customerName": "Jane Doe",
    "dnis": "Desktop",
    "entryPointId": "AXCqFyz1G2pKap9PI-mW",
    "guestId":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1MilkZjM3LTRlODEtOGMxMilmZjkyMTE3MmQ1NzY",
    "mediaChannel": "web",
    "pauseDuration": "10",
    "pauseResumeEnabled": "true",
    "reason": "test",
    "reasonCode": "Sales",
    "ronaTimeout": "30",
    "roomTitle": "Help Jane Doe with Sales",
    "taskToBeSelfServiced": "false",
    "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
    "templateName": "Desktop",
    "tenantId": "133",
    "virtualTeamName": "Chat_Queue",
    "vteamId": "AXCqFyz1G2pKap9PI-mW"
},
"contactDirection": {
    "type": "INBOUND"
},
"currentVTeam": "3270",
"interactionId": "5622d599-40af-11eb-8606-657fd6cc0548",
"isFcManaged": false,
"isTerminated": false,
"media": {
    "Y21zY29zcGFyazovL3VzL1JPT00vNjJkYjBmMzAtNDBhZi0xMWViLWE3ZGItYjkyYTZkMmI5NTI1": {
        "holdTimestamp": null,
        "isHold": false,
        "mType": "mainCall",
        "mediaMgr": "cmm",
        "mediaResourceId":
"Y21zY29zcGFyazovL3VzL1JPT00vNjJkYjBmMzAtNDBhZi0xMWViLWE3ZGItYjkyYTZkMmI5NTI1",
        "mediaType": "chat",
        "participants": ["janedoe@gmail.com",
"7d12d9ea-e8e0-41ee-81bf-c11a685b64ed"]
    },

```



```

"consult_Y21zY29zcGFyazovL3VzL1JPT00vNjJkYjBmMzAtNDBhZi0xMWViLWE3ZGItYjkyYTZkMmI5NTI1": {
  "holdTimestamp": null,
  "isHold": false,
  "mType": "consult",
  "mediaMgr": "cmm",
  "mediaResourceId":
"consult_Y21zY29zcGFyazovL3VzL1JPT00vNjJkYjBmMzAtNDBhZi0xMWViLWE3ZGItYjkyYTZkMmI5NTI1",
  "mediaType": "chat",
  "participants": ["aad323de-32d8-48c9-af6b-b68dfbabfe20",
"7d12d9ea-e8e0-41ee-81bf-c11a685b64ed"]
}
},
"mediaChannel": "web",
"mediaType": "chat",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"outboundType": null,
"owner": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
"participants": {
  "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed": {
    "channelId": "7272c119-821f-485a-b352-29belf3f0fff",
    "consultState": "consultInitiated",
    "consultTimestamp": 1608240860457,
    "dn": "9997770110",
    "hasJoined": true,
    "id": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "isConsulted": false,
    "isWrapUp": false,
    "joinTimestamp": 1608240840633,
    "lastUpdated": 1608240840633,
    "name": "uui-agent4 uui-agent4",
    "pType": "Agent",
    "queueId": "3270",
    "queueMgrId": "aqm",
    "sessionId": "b8c096ab-f9a0-49be-9efd-f8c78e1a7061",
    "siteId": "473",
    "teamId": "962",
    "teamName": "Team X",
    "type": "Agent",
    "wrapUpTimestamp": null
  },
  "aad323de-32d8-48c9-af6b-b68dfbabfe20": {
    "channelId": "b1da7e42-2aeb-492b-ad06-115ba8dfa506",
    "consultState": null,
    "consultTimestamp": null,
    "dn": "9997770109",
    "hasJoined": false,
    "id": "aad323de-32d8-48c9-af6b-b68dfbabfe20",
    "isConsulted": true,
    "isWrapUp": false,
    "joinTimestamp": null,
    "lastUpdated": 1608240860436,
    "name": "uui-agent2 uui-agent2",
    "pType": "Agent",
    "queueId": "3270",
    "queueMgrId": "aqm",
    "sessionId": "3443f9d7-68de-42a0-a590-176a37a3565a",
    "siteId": "473",
    "teamId": "962",
    "teamName": "Team X",
    "type": "Agent",
    "wrapUpTimestamp": null
  },
  "janedoe@gmail.com": {

```

```

        "id": "janedoe@gmail.com",
        "pType": "Customer",
        "type": "Customer"
      }
    },
    "previousVTeams": [],
    "state": "consult"
  },
  "interactionId": "5622d599-40af-11eb-8606-657fd6cc0548",
  "mediaResourceId":
  "Y21zY29zcGFyazovL3VzL1JPT00vNjJkYjBmMzAtNDBhZi0xMWViLWE3ZGItYjkyYTZkMmI5NTI1",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "9f682870-40af-11eb-bc9c-df3ff755538c",
  "type": "AgentConsultCreated"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_6dc77353-e810-4751-82ac-dab03e3951c2",
"type": "RoutingMessage"
}

```

consultConference()

Initiates a conference request with one of the agents from the list of buddyAgents.

Example

```

await Desktop.agentContact.consultConference({
  interactionId: "a26d035d - b547 - 11 ea - 984 b - d3128111ce6b",
  data: {
    agentId: "43 bfcdef - 5551 - 45 b1 - a8f9 - a0b33e66e3fe",
    destAgentId: "f795f41f - 3782 - 44 fa - 97 a4 - c8b4dc029477",
    mediaType: "chat"
  }
});
// consult conference Payload Type

type ConsultConferencePayload = {
  interactionId: string; data: ConsultPayload | ConsultDN
}

type ConsultDN = {
  destAgentId: string;
  destinationType: string;
  mediaType: string;
  trackingId ? : string | undefined;
}

type ConsultPayload = {
  agentId: string;
  destAgentId: string | undefined;
  mediaType ? : string | undefined;
}

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the conference request.	Yes

Name	Type	Description	Required
-->agentId	String	Unique identifier of the agent.	Yes
-->destAgentId	String	Unique identifier of the conference destination agent.	Yes
-->mediaType	String	The media channel type such as chat.	Yes
-->destinationType	String	The type of consult conference to place. The destination type can be Agent, Queue, or DN.	Yes
-->trackingId	String	Unique identifier based on the conference type.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentConsultConferenced`; else, `AgentConsultConferenceFailed`.

Example Response

```
const consultConferenceResponse = {
  "data": {
    "agentId": "337c1cca-d79b-44fc-8d8e-20d1b5e4dadf",
    "consultMediaResourceId":
"consult_Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1",
    "destAgentId": "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "agentEditable": false,
          "displayName": "ani",
          "name": "ani",
          "type": "STRING",
          "value": "displayName"
        },
        "appUser": {
          "agentEditable": false,
          "displayName": "appUser",
          "name": "appUser",
          "type": "STRING",
          "value": "appuser-1212"
        },
        "customerName": {
          "agentEditable": false,
          "displayName": "customerName",
          "name": "customerName",
          "type": "STRING",
          "value": "displayName"
        },
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",

```

consultConference()

```

        "value": "AXVP2lxFOG6K4V3pFKRG"
    },
    "ronaTimeout": {
        "agentEditable": false,
        "displayName": "ronaTimeout",
        "name": "ronaTimeout",
        "type": "STRING",
        "value": "30"
    },
    "virtualTeamName": {
        "agentEditable": false,
        "displayName": "virtualTeamName",
        "name": "virtualTeamName",
        "type": "STRING",
        "value": "QA-e2e-SocialQueue"
    }
},
"callAssociatedDetails": {
    "ani": "*****",
    "appUser": "appuser-1212",
    "customerName": "displayName",
    "dn": "*****",
    "ronaTimeout": "30",
    "virtualTeamName": "QA-e2e-SocialQueue"
},
"callFlowParams": {},
"callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "5134",
    "ani": "*****",
    "appUser": "appuser-1212",
    "customerName": "displayName",
    "dnis": "*****",
    "pauseDuration": "30",
    "pauseResumeEnabled": "true",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "82",
    "virtualTeamName": "QA-e2e-SocialQueue",
    "vteamId": "5134"
},
"contactDirection": {
    "type": "INBOUND"
},
"currentVTeam": "5134",
"interactionId": "1da228dc-66cc-11eb-b852-ed71847d1aec",
"isFcManaged": false,
"isTerminated": false,
"media": {

    "Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1": {
        "holdTimestamp": null,
        "isHold": false,
        "mType": "mainCall",
        "mediaMgr": "socialmm",
        "mediaResourceId":

        "Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1",
        "mediaType": "social",
        "participants": [
            "displayName",
            "337clcca-d79b-44fc-8d8e-20d1b5e4dadf",
            "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2"
        ]
    },

```

```

"consult_Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1": {
    "holdTimestamp": null,
    "isHold": false,
    "mType": "consult",
    "mediaMgr": "socialmm",
    "mediaResourceId":
"consult_Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1",
    "mediaType": "social",
    "participants": [
        "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2",
        "337c1cca-d79b-44fc-8d8e-20d1b5e4dadf",
        "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2"
    ]
},
"mediaChannel": "type",
"mediaType": "social",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"outboundType": null,
"owner": "337c1cca-d79b-44fc-8d8e-20d1b5e4dadf",
"participants": {
    "337c1cca-d79b-44fc-8d8e-20d1b5e4dadf": {
        "channelId": "e08a6068-688d-472b-887b-1c5d39c7155f",
        "consultState": "conferencing",
        "consultTimestamp": 1612431260416,
        "dn": "*****",
        "hasJoined": true,
        "id": "337c1cca-d79b-44fc-8d8e-20d1b5e4dadf",
        "isConsulted": false,
        "isWrapUp": false,
        "joinTimestamp": 1612431253790,
        "lastUpdated": 1612431253790,
        "name": "social-agent2",
        "pType": "Agent",
        "queueId": "5134",
        "queueMgrId": "aqm",
        "sessionId": "a9aa9867-e62f-4415-9ea4-6babace39d4f",
        "siteId": "205",
        "teamId": "384",
        "teamName": "TeamSales",
        "type": "Agent",
        "wrapUpTimestamp": null
    },
    "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2": {
        "channelId": "841235e4-7052-455b-be64-30f2f3c4a59b",
        "consultState": "conferencing",
        "consultTimestamp": 1612431260416,
        "dn": "*****",
        "hasJoined": true,
        "id": "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2",
        "isConsulted": true,
        "isWrapUp": false,
        "joinTimestamp": 1612431258137,
        "lastUpdated": 1612431258137,
        "name": "social-agent1",
        "pType": "Agent",
        "queueId": "5134",
        "queueMgrId": "aqm",
        "sessionId": "600e7ebb-20aa-4c91-a4f0-ebc596249603",
        "siteId": "205",
        "teamId": "384",
        "teamName": "TeamSales",
        "type": "Agent",
    }
}

```

```

        "wrapUpTimestamp": null
      },
      "displayName": {
        "id": "displayName",
        "pType": "Customer",
        "type": "Customer"
      }
    },
    "previousVTeams": [
      "AXVP2lxFOG6K4V3pFKRG"
    ],
    "state": "conference",
    "workflowManager": null
  },
  "interactionId": "1da228dc-66cc-11eb-b852-ed71847d1aec",
  "mediaResourceId":
"Y21zY29zcGFyYXovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "4a958ace-49a2-49aa-b2e5-8798ede03f13",
  "type": "AgentConsultConferenced"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_68c519c8-04f5-48b4-a5cf-609a09ccd765",
"type": "RoutingMessage"
}

```

consultEnd()

Ends a consult request.

Example

```

await Desktop.agentContact.consultEnd({
  interactionId: "a26d035d - b547 - 11 ea - 984 b - d3128111ce6b",
  isConsult: false
});

// Consult end payload type
{
  interactionId: string;isConsult: boolean
}

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
isConsult	Boolean	Determines whether the task request is consult or not. <ul style="list-style-type: none"> • True—The task request is consult. • False—The task request is not consult. 	Yes

Returns

{Object} The object with the retrieved data.



Note The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, and `isConsult` is *true*, the value is `AgentConsultEnded`; else, `AgentConsultConferenceEnded`.

If the method is unsuccessful, the value is `AgentConsultEndFailed`.

Example Response

```
const consultEndResponse = {
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "consultMediaResourceId": "f23b5b8a-4b02-451e-b9f5-8517b5bef97c",
    "destAgentId": "9fca6158-a8bd-43e6-aebe-e5055aea08c5",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "8895579172"
        },
        "ronaTimeout": {
          "agentEditable": false,
          "displayName": "ronaTimeout",
          "name": "ronaTimeout",
          "type": "STRING",
          "value": "30"
        },
        "virtualTeamName": {
          "agentEditable": false,
          "displayName": "virtualTeamName",
          "name": "virtualTeamName",
          "type": "STRING",
          "value": "Outdial Queue-1"
        }
      },
      "callAssociatedDetails": {
        "dn": "8895579172",
        "ronaTimeout": "30",
        "virtualTeamName": "Outdial Queue-1"
      },
      "callFlowParams": {
        "OutdialQueue": {
          "description": "(vteam, The Outdial Queue.)",
          "name": "OutdialQueue",
          "qualifier": "vteam",
          "value": "3264",
          "valueDataType": "string"
        }
      },
      "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3264",
        "dnis": "8895579172",
        "isPaused": "false",
        "outdialTransferToQueueEnabled": "false",
        "pauseDuration": "10",
        "pauseResumeEnabled": "true",
```

consultEnd()

```

        "recordInProgress": "true",
        "ronaTimeout": "30",
        "taskToBeSelfServiced": "false",
        "tenantId": "133",
        "virtualTeamName": "Outdial Queue-1",
        "vteamId": "AXCLfZZU9S1oTdqE1OFZ"
    },
    "contactDirection": {
        "type": "OUTBOUND"
    },
    "currentVTeam": "3264",
    "interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
    "isFcManaged": false,
    "isTerminated": false,
    "media": {
        "1dd08726-bca5-4ce8-9b51-b20dca8ab154": {
            "holdTimestamp": 1612356600241,
            "isHold": true,
            "mType": "mainCall",
            "mediaMgr": "vmm",
            "mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
            "mediaType": "telephony",
            "participants": [
                ""
            ]
        }
    },
    "mediaChannel": "dialer",
    "mediaType": "telephony",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": "OUTDIAL",
    "owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "participants": {
        "": {
            "id": "",
            "pType": "Customer",
            "type": "Customer"
        },
        "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
            "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
            "consultState": null,
            "consultTimestamp": null,
            "dn": "8895579172",
            "hasJoined": true,
            "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
            "isConsulted": false,
            "isWrapUp": false,
            "joinTimestamp": 1612354635705,
            "lastUpdated": 1612354635705,
            "name": "John Doe",
            "pType": "Agent",
            "queueId": "3264",
            "queueMgrId": "aqm",
            "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
            "siteId": "472",
            "teamId": "960",
            "teamName": "Email_Team",
            "type": "Agent",
            "wrapUpTimestamp": null
        }
    },
    "previousVTeams": [
    ],

```



```

        "state": "connected",
        "workflowManager": null
      },
      "interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
      "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
      "queueMgr": "aqm",
      "trackingId": "f7c7ecd2-ef27-4273-bc11-1543bddc8810",
      "type": "AgentConsultEnded"
    },
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "trackingId": "notifs_860d1e16-88b8-4d6c-8bac-18075d72a7e8",
    "type": "RoutingMessage"
  }
}

```

decline()

Declines a consult request.

Example

```

await Desktop.agentContact.decline({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3";
  data: {
    mediaResourceId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
  }
  isConsult: true;
});

// Decline payload type
{
  interactionId: string;
  data: Contact.declinePayload;
  isConsult: boolean
}

type declinePayload = {
  mediaResourceId: string;
}

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the consult request.	Yes
-->mediaResourceId	String	Unique identifier of the media resource.	Yes
isConsult	Boolean	Determines whether the task request is consult or not. <ul style="list-style-type: none"> • True—The task request is consult. • False—The task request is not consult. 	Yes

Returns

{Object} The object with the retrieved data.

**Note**

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, and `isConsult` is *true*, the value is `AgentConsultFailed`; else, `AgentOfferContactRona`.

Example Response

```
const declineResponse = {
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "destAgentId": "f795f41f-3782-44fa-97a4-c8b4dc029477",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {},
      "callAssociatedDetails": {},
      "callFlowParams": {},
      "callProcessingDetails": {},
      "contactDirection": {
        "type": "INBOUND"
      },
      "currentVTeam": "",
      "interactionId": "9ad96648-415b-11eb-881c-ed6dd0d45a04",
      "isFcManaged": false,
      "isTerminated": false,
      "media": {},
      "mediaChannel": "none",
      "mediaType": "none",
      "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
      "outboundType": null,
      "owner": null,
      "participants": {},
      "previousVTeams": [],
      "state": "none",
      "workflowManager": null
    },
    "interactionId": "a26d035d-b547-11ea-984b-d3128111ce6b",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "queueMgr": "aqm",
    "reason": "Test",
    "reasonCode": 500,
    "trackingId": "2037cb70-653d-11eb-8a11-a3c35f103d4b",
    "type": "AgentConsultFailed"
  },
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "trackingId": "notifs_a241b1c4-3d42-4ad0-8738-71b1e7611550",
  "type": "RoutingMessage"
}
```

cancelCtq()

Cancels a consult to queue request.

Example

```
await Desktop.agentContact.cancelCtq({
  interactionId: "a26d035d - b547 - 11 ea - 984 b - d3128111ce6b",
  data: {
    agentId: "f795f41f - 3782 - 44 fa - 97 a4 - c8b4dc029477",
    queueId: "3268"
  }
})
```

```
});

// cancelCtq Payload
{
  interactionId: string; data: Contact.cancelCtq
}
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the consult to queue request.	Yes
-->agentId	String	Unique identifier of the agent.	Yes
-->queueId	String	Unique identifier of the queue.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentCtqCancelled`; else, `AgentCtqCancelFailed`.

Example Response

```
const cancelCtqResponse = {
  "data": {
    "agentId": "a96ffe92-f5c8-4715-9427-fa32a7e57ccc",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {},
      "callAssociatedDetails": {},
      "callFlowParams": {},
      "callProcessingDetails": {},
      "contactDirection": {
        "type": "INBOUND"
      },
      "currentVTeam": "",
      "interactionId": "9ad96648-415b-11eb-881c-ed6dd0d45a04",
      "isFcManaged": false,
      "isTerminated": false,
      "media": {},
      "mediaChannel": "none",
      "mediaType": "none",
      "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
      "outboundType": null,
      "owner": null,
      "participants": {},
      "previousVTeams": [],
      "state": "none"
    },
    "interactionId": "c8e35eec-a977-409b-922d-7d2825c63cf2",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "queueId": "7352",
    "queueMgr": "aqm",
    "trackingId": "ee2bc7e0-50c8-11eb-a1f8-5d0ad357b8a5",
```

```

        "type": "AgentCtqCancelled"
    },
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "trackingId": "notifs_908141bb-3f1f-43fe-b7e2-5a132c97745f",
    "type": "RoutingMessage"
}

```

wrapup()

Wraps up work for a task. Wrap up reason is applied after the task has been ended either by the agent or by the customer. A wrap up is essential and is the last step in an agent's flow of handling tasks.

Example

```

await Desktop.agentContact.wrapup({
  interactionId: "dab1a2f0 - bad0 - 11 ea - 8e7 c - 0 d99ede2535a",
  data: {
    wrapUpReason: "Example reason here";
    auxCodeId: "0";
    isAutoWrapup: "false";
  }
});

// Wrap Payload Type

{
  interactionId: string; data: Contact.WrapupPayLoad
}

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the wrap up.	Yes
-->wrapUpReason	String	Wrap up reason indicating why a customer called the contact center.	Yes
-->auxCodeId	String	Unique identifier of the agent state.	Yes
-->isAutoWrapup	Boolean	Determines whether the auto wrap up is enabled or not. True—The auto wrap up is enabled. False—The auto wrap up is disabled.	

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentWrappedUp`; else, `AgentWrapupFailed`.

Example Response

```

const wrapUpResponse = {
  "data": {
    "agentId": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "value": "janedoe@gmail.com"
        },
        "category": {
          "value": "Sales"
        },
        "customerName": {
          "value": "Jane Doe"
        },
        "dn": {
          "value": "Desktop"
        },
        "entryPointId": {
          "value": "AXCqFyz1G2pKap9PI-mW"
        },
        "guestId": {
          "value": "Y21zY29zcGFyazovL3VzL1BFT1BMRS81NzNiODg1M1kZjM3LTRlODEtOGMxM1mZjkyMTE3MmQ1NzY"
        },
        "mediaChannel": {
          "value": "web"
        },
        "reason": {
          "value": "Test"
        },
        "reasonCode": {
          "value": "Sales"
        },
        "ronaTimeout": {
          "value": "30"
        },
        "roomTitle": {
          "value": "Help Jane Doe with Sales"
        },
        "taskToBeSelfServiced": {
          "value": "false"
        },
        "templateId": {
          "value": "b57990c0-5ec6-11ea-96ee-5df5aef56329"
        },
        "templateName": {
          "value": "Desktop"
        },
        "virtualTeamName": {
          "value": "Chat_Queue"
        }
      },
      "callAssociatedDetails": {
        "ani": "janedoe@gmail.com",
        "category": "Sales",
        "customerName": "Jane Doe",
        "dn": "Desktop",
        "entryPointId": "AXCqFyz1G2pKap9PI-mW",
        "guestId": "Y21zY29zcGFyazovL3VzL1BFT1BMRS81NzNiODg1M1kZjM3LTRlODEtOGMxM1mZjkyMTE3MmQ1NzY",
        "mediaChannel": "web",
        "reason": "Test",

```

```

        "reasonCode": "Sales",
        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Sales",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "virtualTeamName": "Chat_Queue"
    },
    "callFlowParams": {
        "Automation": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Automation",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Debit": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Debit",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Sales": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Sales",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        }
    },
    "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3270",
        "ani": "janedoe@gmail.com",
        "category": "Sales",
        "customerName": "Jane Doe",
        "dnis": "Desktop",
        "entryPointId": "AXCqFyz1G2pKap9PI-mW",
        "guestId":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1Mi1kZjM3LTRlODEtOGMxMi1mZjkyMTE3MmQ1NzY",
        "mediaChannel": "web",
        "pauseDuration": "10",
        "pauseResumeEnabled": "true",
        "reason": "Test",
        "reasonCode": "Sales",
        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Sales",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "tenantId": "133",
        "virtualTeamName": "Chat_Queue",
        "vteamId": "AXCqFyz1G2pKap9PI-mW"
    },
    "contactDirection": {
        "type": "INBOUND"
    },
    "currentVTeam": "3270",
    "interactionId": "a66d05ab-40ab-11eb-b59a-7d3326b3ffc1",
    "isFcManaged": false,
    "isTerminated": true,
    "media": {},
    "mediaChannel": "web",

```

```

    "mediaType": "chat",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": null,
    "owner": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "participants": {
      "janedoe@gmail.com": {
        "id": "janedoe@gmail.com",
        "pType": "Customer",
        "type": "Customer"
      }
    },
    "previousVTeams": [],
    "state": "connected"
  },
  "interactionId": "a66d05ab-40ab-11eb-b59a-7d3326b3ffc1",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "5272a610-40ac-11eb-8866-d7ef6ff89aa6",
  "type": "AgentWrappedUp",
  "wrapUpAuxCodeId": "0"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_b5e0ebb7-fd1d-479f-8663-81a64030cee6",
"type": "RoutingMessage"
}

```

vTeamList()

Retrieves a list of vTeams (queues) available for transfer. Instead of transferring your interaction to a particular agent, you can transfer it to an appropriate queue (vTeam).

Example

```

await Desktop.agentContact.vteamList({
  data: {
    agentProfileId: "AXClfZhH9S1oTdqE1OFw",
    agentSessionId: "5 a84d32c - 691 b - 4500 - b163 - d6cdba2a3163",
    channelType: "chat",
    type: "inboundqueue"
    trackingId: "9c9b97e0-4194-11eb-b819-7b8dd50ee0cd",
  }
});

// Vteam List Payload Type
{
  data: Contact.VTeam
}

```

The following table lists the payload details:

Name	Type	Description	Required
data	Object	Options of the list of queues available for transfer.	Yes
-->agentProfileId	String	Unique identifier of the agent profile.	Yes
-->agentSessionId	String	Unique identifier of the agent session.	Yes
-->channelType	String	The media channel type such as chat.	Yes
-->type	String	The type of queue list.	Yes

Name	Type	Description	Required
-->trackingId	String	Unique identifier based on the queue list available for transfer.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `VteamList`; else, `VteamListFailed`.

Example Response

```
const vTeamResponse = {
  "data": {
    "agentSessionId": "5a84d32c-691b-4500-b163-d6cdba2a3163",
    "callData": "",
    "data": {
      "allowConsultToQueue": true,
      "vteamList": [{
        "analyzerId": "AXT3_BAV0G6K4V3p9euK",
        "callDistributionGroups": [],
        "channelType": "chat",
        "id": "7178",
        "maxTimeInQueue": 500,
        "name": "Chat_Demo",
        "routingType": null,
        "skillBasedRoutingType": null,
        "type": "inboundqueue"
      }, {
        "analyzerId": "AXNSD0dSQJBc-emowhp3",
        "callDistributionGroups": [],
        "channelType": "chat",
        "id": "5900",
        "maxTimeInQueue": 500,
        "name": "Chat_Automation_Queue",
        "routingType": null,
        "skillBasedRoutingType": null,
        "type": "inboundqueue"
      }, {
        "analyzerId": "AXDiuOLjKin42Ov3QJXN",
        "callDistributionGroups": [],
        "channelType": "chat",
        "id": "3381",
        "maxTimeInQueue": 500,
        "name": "Dont_delete_Chat_queue",
        "routingType": null,
        "skillBasedRoutingType": null,
        "type": "inboundqueue"
      }, {
        "analyzerId": "AXCZ1Cc0B023QCbPLXpr",
        "callDistributionGroups": [],
        "channelType": "chat",
        "id": "3270",
        "maxTimeInQueue": 500,
        "name": "Chat_Queue",
        "routingType": null,
        "skillBasedRoutingType": null,

```



```

        "type": "inboundqueue"
    }]
  },
  "jsMethod": "vteamListChanged"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_0e6eff89-165e-46b6-8c85-1aada01dab82",
"type": "VteamList"
}

```

vTeamTransfer()

Initiates a vTeam transfer request.

Example

```

await Desktop.agentContact.vteamTransfer({
  interactionId: "dabla2f0 - bad0 - 11 ea - 8e7 c - 0 d99ede2535a",
  data: {
    vteamId: "AXDiuOLjKin42Ov3QJXN",
    vteamType: "inboundqueue"
  }
});

// Vteam Transfer Payload Type

{
  interactionId: string;
  data: Contact.vteamTransferPayLoad
}

type vteamTransferPayLoad = {
  vteamId: string;
  vteamType: string;
}

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the vTeam transfer request.	Yes
-->vteamId	String	Unique identifier of the vTeam.	Yes
-->vteamType	String	The type of vTeam.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentVteamTransferred`; else, `AgentVteamTransferFailed`.

Example Response

```

const vTeamTransferResponse = {
  "agentId": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
  "eventType": "RoutingMessage",
  "interaction": {
    "callAssociatedData": {
      "ani": {
        "value": "janedoe@gmail.com"
      },
      "category": {
        "value": "Sales"
      },
      "customerName": {
        "value": "Jane Doe"
      },
      "dn": {
        "value": "Desktop"
      },
      "entryPointId": {
        "value": "AXCqFyz1G2pKap9PI-mW"
      },
      "guestId": {
        "value":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1Mi1kZjM3LTRlODEtOGMxMi1mZjkyMTE3MmQ1NzY"
      },
      "mediaChannel": {
        "value": "web"
      },
      "reason": {
        "value": "Test"
      },
      "reasonCode": {
        "value": "Sales"
      },
      "ronaTimeout": {
        "value": "30"
      },
      "roomTitle": {
        "value": "Help Jane Doe with Sales"
      },
      "taskToBeSelfServiced": {
        "value": "false"
      },
      "templateId": {
        "value": "b57990c0-5ec6-11ea-96ee-5df5aef56329"
      },
      "templateName": {
        "value": "Desktop"
      },
      "virtualTeamName": {
        "value": "Chat_Queue"
      }
    },
    "callAssociatedDetails": {
      "ani": "janedoe@gmail.com",
      "category": "Sales",
      "customerName": "Jane Doe",
      "dn": "Desktop",
      "entryPointId": "AXCqFyz1G2pKap9PI-mW",
      "guestId":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS81NzNiODg1Mi1kZjM3LTRlODEtOGMxMi1mZjkyMTE3MmQ1NzY",
      "mediaChannel": "web",
      "reason": "Test",
      "reasonCode": "Sales",
      "ronaTimeout": "30",

```

```

        "roomTitle": "Help Jane Doe with Sales",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "virtualTeamName": "Chat_Queue"
    },
    "callFlowParams": {
        "Automation": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Automation",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Debit": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Debit",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Sales": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Sales",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        }
    },
    "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3270",
        "ani": "janedoe@gmail.com",
        "category": "Sales",
        "customerName": "Jane Doe",
        "dnis": "Desktop",
        "doNotRouteToAgents": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
        "entryPointId": "AXCqFyz1G2pKap9PI-mW",
        "guestId":
"Y21zy29zcGFyazovL3VzL1BFTT1BMRS81NzNiODg1MilkZjM3LTRlODEtOGMxMilmZjkyMTE3MmQ1NzY",
        "mediaChannel": "web",
        "pauseDuration": "10",
        "pauseResumeEnabled": "true",
        "reason": "Test",
        "reasonCode": "Sales",
        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Sales",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "tenantId": "133",
        "virtualTeamName": "Chat_Queue",
        "vteamId": "7178",
        "vteamType": "inboundqueue"
    },
    "contactDirection": {
        "type": "INBOUND"
    },
    "currentVTeam": "7178",
    "interactionId": "29bafa9c-40b5-11eb-8606-edd437f1b927",
    "isFcManaged": false,
    "isTerminated": false,
    "media": {

```

blindTransfer()

```

"Y21zY29zcGFyazovL3VzL1JPT00vMmNiMTMwMDAtNDBiNS0xMWViLWE4M2ItZWl4ZTQwMDk1MDk5": {
    "holdTimestamp": null,
    "isHold": false,
    "mType": "mainCall",
    "mediaMgr": "cmm",
    "mediaResourceId":
"Y21zY29zcGFyazovL3VzL1JPT00vMmNiMTMwMDAtNDBiNS0xMWViLWE4M2ItZWl4ZTQwMDk1MDk5",
    "mediaType": "chat",
    "participants": ["janedoe@gmail.com",
"7d12d9ea-e8e0-41ee-81bf-c11a685b64ed"]
    },
    "mediaChannel": "web",
    "mediaType": "chat",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": null,
    "owner": null,
    "participants": {
        "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed": {
            "channelId": "ccbffa6f-4322-4e06-850f-e0ab2915f238",
            "consultState": null,
            "consultTimestamp": null,
            "dn": "9997770110",
            "hasJoined": true,
            "id": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
            "isConsulted": false,
            "isWrapUp": true,
            "joinTimestamp": 1608243279477,
            "lastUpdated": 1608254153975,
            "name": "FirstName LastName",
            "pType": "Agent",
            "queueId": "3270",
            "queueMgrId": "aqm",
            "sessionId": "b8c096ab-f9a0-49be-9efd-f8c78e1a7061",
            "siteId": "473",
            "teamId": "962",
            "teamName": "Team X",
            "type": "Agent",
            "wrapUpTimestamp": 1608254153975
        },
        "janedoe@gmail.com": {
            "id": "janedoe@gmail.com",
            "pType": "Customer",
            "type": "Customer"
        }
    },
    "previousVTeams": ["3270"],
    "state": "new"
},
"interactionId": "29bafa9c-40b5-11eb-8606-edd437f1b927",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"trackingId": "91480930-40ce-11eb-840f-7f9e29c6b481",
"type": "AgentVteamTransferred"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25", "trackingId":
"notifs_08102baa-3437-4c6d-812c-db4ce8755ea0", "type": "RoutingMessage"
}

```

blindTransfer()

Initiates a blind transfer request.

Example

```
await Desktop.agentContact.blindTransfer({
  interactionId: "08259 a2b - bacf - 11 ea - bdd9 - abbd16eaf1d5",
  data: {
    agentId: "f795f41f - 3782 - 44 fa - 97 a4 - c8b4dc029477",
    destAgentId: "299 b728a - d6f8 - 4934 - 8 fad - 577525 c0b7fc",
    mediaType: "chat",
    destAgentTeamId: "964",
    destAgentDN: "9997770095",
    destSiteId: "472"
  }
});

// BlindTransfer Payload Type

{
  interactionId: string; data: Contact.blindTransferPayload
}

type blindTransferPayload = {
  agentId: string;
  destAgentId: string;
  mediaType: string;
  destAgentTeamId: string;
  destAgentDN: string;
  destSiteId: string;
}
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the blind transfer request.	Yes
-->agentId	String	Unique identifier of the agent.	Yes
-->destAgentId	String	Unique identifier of the blind transfer request destination agent.	Yes
-->mediaType	String	The media channel type such as chat.	Yes
-->destAgentTeamId	String	Unique identifier of the team to which the destination agent belongs.	Yes
-->destAgentDN	String	The dial number of the destination agent.	Yes
-->destSiteId	String	Unique identifier of the destination site.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentBlindTransferred`; else, `AgentBlindTransferFailedEvent`.

Example Response

```
const blindTransferResponse = {
  "data": {
    "agentId": "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2",
    "destAgentId": "337c1cca-d79b-44fc-8d8e-20d1b5e4dadf",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "agentEditable": false,
          "displayName": "ani",
          "name": "ani",
          "type": "STRING",
          "value": "displayName"
        },
        "appUser": {
          "agentEditable": false,
          "displayName": "appUser",
          "name": "appUser",
          "type": "STRING",
          "value": "qaus1-appuser-1212"
        },
        "customerName": {
          "agentEditable": false,
          "displayName": "customerName",
          "name": "customerName",
          "type": "STRING",
          "value": "displayName"
        },
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "AXVP21xF0G6K4V3pFKRG"
        },
        "ronaTimeout": {
          "agentEditable": false,
          "displayName": "ronaTimeout",
          "name": "ronaTimeout",
          "type": "STRING",
          "value": "30"
        },
        "virtualTeamName": {
          "agentEditable": false,
          "displayName": "virtualTeamName",
          "name": "virtualTeamName",
          "type": "STRING",
          "value": "QA-e2e-SocialQueue"
        }
      },
      "callAssociatedDetails": {
        "ani": "*****",
        "appUser": "appuser-1212",
        "customerName": "displayName",
        "dn": "*****",
        "ronaTimeout": "30",
        "virtualTeamName": "QA-e2e-SocialQueue"
      },
      "callFlowParams": {},
      "callProcessingDetails": {
        "BLIND_TRANSFER_IN_PROGRESS": "true",
```

```

        "QMgrName": "aqm",
        "QueueId": "5134",
        "ani": "*****",
        "appUser": "gaus1-appuser-1212",
        "customerName": "displayName",
        "dnis": "*****",
        "pauseDuration": "30",
        "pauseResumeEnabled": "true",
        "ronaTimeout": "30",
        "taskToBeSelfServiced": "false",
        "tenantId": "82",
        "virtualTeamName": "SocialQueue",
        "vteamId": "5134"
    },
    "contactDirection": {
        "type": "INBOUND"
    },
    "currentVTeam": "5134",
    "interactionId": "1da228dc-66cc-11eb-b852-ed71847dlaec",
    "isFcManaged": false,
    "isTerminated": false,
    "media": {
        "Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1": {
            "holdTimestamp": null,
            "isHold": false,
            "mType": "mainCall",
            "mediaMgr": "socialmm",
            "mediaResourceId":
        "Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1",
            "mediaType": "social",
            "participants": [
                "displayName"
            ]
        }
    },
    "mediaChannel": "type",
    "mediaType": "social",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": null,
    "owner": "337clcca-d79b-44fc-8d8e-20d1b5e4dadf",
    "participants": {
        "337clcca-d79b-44fc-8d8e-20d1b5e4dadf": {
            "channelId": "e08a6068-688d-472b-887b-1c5d39c7155f",
            "consultState": null,
            "consultTimestamp": null,
            "dn": "*****",
            "hasJoined": false,
            "id": "337clcca-d79b-44fc-8d8e-20d1b5e4dadf",
            "isConsulted": false,
            "isWrapUp": false,
            "joinTimestamp": null,
            "lastUpdated": 1612431250963,
            "name": "social-agent2",
            "pType": "Agent",
            "queueId": "5134",
            "queueMgrId": "aqm",
            "sessionId": "a9aa9867-e62f-4415-9ea4-6babace39d4f",
            "siteId": "205",
            "teamId": "384",
            "teamName": "TeamSales",
            "type": "Agent",
            "wrapUpTimestamp": null
        }
    },

```

```

        "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2": {
            "channelId": "59826e62-08c8-41e6-b51c-4a2b6d81d2f6",
            "consultState": null,
            "consultTimestamp": null,
            "dn": "*****",
            "hasJoined": true,
            "id": "cdf786ef-3285-4b2f-8ef9-3cf8a4c755c2",
            "isConsulted": false,
            "isWrapUp": true,
            "joinTimestamp": 1612431249042,
            "lastUpdated": 1612431251178,
            "name": "gaus1-gt-social-agent1 gaus1-gt-social-agent1",
            "pType": "Agent",
            "queueId": "5134",
            "queueMgrId": "aqm",
            "sessionId": "600e7ebb-20aa-4c91-a4f0-ebc596249603",
            "siteId": "205",
            "teamId": "384",
            "teamName": "TeamSales",
            "type": "Agent",
            "wrapUpTimestamp": 1612431251178
        },
        "displayName": {
            "id": "displayName",
            "pType": "Customer",
            "type": "Customer"
        }
    },
    "previousVTeams": [
        "AXVP2lxFOG6K4V3pFKRG"
    ],
    "state": "new",
    "workflowManager": null
},
"interactionId": "1da228dc-66cc-11eb-b852-ed71847d1aec",
"mediaResourceId":
"Y21zY29zcGFyazovL3VzL1JPT00vNGQ5NGJmMjAtYjYwZi0xMWVhLTk0YjAtNzdjZDBkYWVjNDA1",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"trackingId": "4a958ace-49a2-49aa-b2e5-8798ede03f13",
"type": "AgentBlindTransferred"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_68c519c8-04f5-48b4-a5cf-609a09ccd765",
"type": "RoutingMessage"
}

```

consultTransfer()

Initiates a consult transfer request.

Example

```

await Desktop.agentContact.consultTransfer({
    interactionId: "5 c3d487a - 874 b - 447 d - b5f2 - ce1f626301f5",
    data: {
        agentId: "43 bfcdef - 5551 - 45 b1 - a8f9 - a0b33e66e3fe",
        destAgentId: "f795f41f - 3782 - 44 fa - 97 a4 - c8b4dc029477",
        mediaType: "telephony",
        mediaResourceId: "b102ed10 - fac2 - 4 f8e - bece - 1 c2da6ba6dd8",
        destinationType: "Agent",
    }
});

```



```
// consultTransfer Payload Type

{
  interactionId: string; data: Contact.consultTransferPayload
}

type consultTransferPayload = {
  agentId ? : string | undefined;
  destAgentId: string;
  mediaType: string;
  mediaResourceId: string;
  destinationType ? : string | undefined;
}
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options of the consult transfer request.	Yes
-->agentId	String	Unique identifier of the agent.	Yes
-->destAgentId	String	Unique identifier of the consult transfer request destination agent.	Yes
-->mediaType	String	The media channel type such as telephony.	Yes
-->mediaResourceId	String	Unique identifier of the media resource.	Yes
-->destinationType	String	The type of consult transfer request to be placed. The destination type can be Agent, Queue, or DN.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentConsultTransferred`; else, `AgentConsultTransferFailed`.

Example Response

```
const consultTransferResponse = {
  "data": {
    "agentId": "ff7dbe19-fa3f-4c63-8a1c-04dcffef8e4f",
    "consultMediaResourceId": "fd0af60c-3250-46c3-96f8-1921f6ecf170",
    "destAgentId": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
    "destinationType": "Agent",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "": {
          "agentEditable": false,
          "displayName": "",
          "name": "",

```

```

        "type": "STRING",
        "value": ""
    },
    "IvrPath": {
        "agentEditable": false,
        "displayName": "IvrPath",
        "name": "IvrPath",
        "type": "STRING",
        "value": "EOI"
    },
    "ani": {
        "agentEditable": false,
        "displayName": "ani",
        "name": "ani",
        "type": "STRING",
        "value": "*****"
    },
    "dn": {
        "agentEditable": false,
        "displayName": "dn",
        "name": "dn",
        "type": "STRING",
        "value": "*****"
    },
    "nextVteamId": {
        "agentEditable": false,
        "displayName": "nextVteamId",
        "name": "nextVteamId",
        "type": "STRING",
        "value": "Queue"
    },
    "pathId": {
        "agentEditable": false,
        "displayName": "pathId",
        "name": "pathId",
        "type": "STRING",
        "value": " StartCall PlayDone"
    },
    "ronaTimeout": {
        "agentEditable": false,
        "displayName": "ronaTimeout",
        "name": "ronaTimeout",
        "type": "STRING",
        "value": "30"
    },
    "virtualTeamName": {
        "agentEditable": false,
        "displayName": "virtualTeamName",
        "name": "virtualTeamName",
        "type": "STRING",
        "value": "Queue-1"
    }
},
"callAssociatedDetails": {
    "": "",
    "IvrPath": "EOI",
    "ani": "*****",
    "dn": "*****",
    "nextVteamId": "Queue",
    "pathId": " StartCall PlayDone",
    "ronaTimeout": "30",
    "virtualTeamName": "Queue-1"
},
"callFlowParams": {

```

```

    "Queue": {
      "description": "(vteam, A valid VTeam.)",
      "name": "Queue",
      "qualifier": "vteam",
      "value": "1336",
      "valueDataType": "string"
    },
    "WelcomePrompt": {
      "description": "(mediaFile, A valid media file.)",
      "name": "WelcomePrompt",
      "qualifier": "mediaFile",
      "value": "GAwelcome.wav",
      "valueDataType": "string"
    }
  },
  "callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "1336",
    "ani": "*****",
    "dnis": "*****",
    "isPaused": "false",
    "jscriptId": "AW7HP3QNB44q0sL-cTVz",
    "pauseDuration": "30",
    "pauseResumeEnabled": "true",
    "recordInProgress": "true",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "82",
    "virtualTeamName": "Queue-1",
    "vteamId": "1336"
  },
  "contactDirection": {
    "type": "INBOUND"
  },
  "currentVTeam": "1336",
  "interactionId": "1277da79-dc49-439c-ab72-f00453a1ff71",
  "isFcManaged": false,
  "isTerminated": false,
  "media": {
    "1277da79-dc49-439c-ab72-f00453a1ff71": {
      "holdTimestamp": null,
      "isHold": false,
      "mType": "mainCall",
      "mediaMgr": "vmm",
      "mediaResourceId": "1277da79-dc49-439c-ab72-f00453a1ff71",
      "mediaType": "telephony",
      "participants": [
        "*****",
        "9b036d89-930c-4187-a5bd-5dbb1439fe41"
      ]
    }
  },
  "mediaChannel": "broadcloud",
  "mediaType": "telephony",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "outboundType": null,
  "owner": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
  "participants": {
    "*****": {
      "id": "*****",
      "pType": "Customer",
      "type": "Customer"
    },
    "9b036d89-930c-4187-a5bd-5dbb1439fe41": {

```

```

        "channelId": "3de6706f-95b7-485a-9304-30928f7ee52e",
        "consultState": null,
        "consultTimestamp": null,
        "dn": "*****",
        "hasJoined": true,
        "id": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
        "isConsulted": true,
        "isWrapUp": false,
        "joinTimestamp": 1612429549032,
        "lastUpdated": 1612429549032,
        "name": "qaus1-gt-user1 qaus1-gt-user1",
        "pType": "Agent",
        "queueId": "1336",
        "queueMgrId": "aqm",
        "sessionId": "90896c91-0f77-4bc6-aec3-488a34ca88d3",
        "siteId": "205",
        "teamId": "598",
        "teamName": "Sales",
        "type": "Agent",
        "wrapUpTimestamp": null
    },
    "ff7dbe19-fa3f-4c63-8a1c-04dcffef8e4f": {
        "channelId": "fb704337-71c3-4802-a03b-84a2d18eeaa0",
        "consultState": "consulting",
        "consultTimestamp": 1612429549052,
        "dn": "*****",
        "hasJoined": true,
        "id": "ff7dbe19-fa3f-4c63-8a1c-04dcffef8e4f",
        "isConsulted": false,
        "isWrapUp": false,
        "joinTimestamp": 1612429537991,
        "lastUpdated": 1612429537991,
        "name": "qaus1-gt-user2 q",
        "pType": "Agent",
        "queueId": "1336",
        "queueMgrId": "aqm",
        "sessionId": "57323f8b-9460-45fa-aaf6-4ca51c327af8",
        "siteId": "205",
        "teamId": "598",
        "teamName": "Sales",
        "type": "Agent",
        "wrapUpTimestamp": null
    }
},
"previousVTeams": [
    "AW6hlAVRrZSGnl185x28"
],
"state": "connected",
"workflowManager": null
},
"interactionId": "1277da79-dc49-439c-ab72-f00453a1ff71",
"mediaResourceId": "1277da79-dc49-439c-ab72-f00453a1ff71",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"trackingId": "05b38615-5bdc-4e1d-8efe-60ealc70fd5c",
"transferredMediaResourceId": "1277da79-dc49-439c-ab72-f00453a1ff71",
"type": "AgentConsultTransferred"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_3464be63-83e7-4b2d-94d5-a63fddc108d5",
"type": "RoutingMessage"
}

```

hold()

Places a voice call on hold.

Example

```
await Desktop.agentContact.hold({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3",
  data: {
    mediaResourceId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
  }
});

// Hold Payload

{
  interactionId: string; data: {
    mediaResourceId: string
  }
}
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options to place the voice call on hold.	Yes
-->mediaResourceId	String	Unique identifier of the media resource.	Yes

Returns

{Object} The object with the retrieved data.



Note The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentContactHeld`; else, `AgentContactHoldFailed`.

Example Response

```
const holdResponse = {
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "destAgentId": null,
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "8895579172"
        },
        "ronaTimeout": {
          "agentEditable": false,
          "displayName": "ronaTimeout",
          "name": "ronaTimeout",

```

hold()

```

        "type": "STRING",
        "value": "30"
    },
    "virtualTeamName": {
        "agentEditable": false,
        "displayName": "virtualTeamName",
        "name": "virtualTeamName",
        "type": "STRING",
        "value": "Outdial Queue-1"
    }
},
"callAssociatedDetails": {
    "dn": "8895579172",
    "ronaTimeout": "30",
    "virtualTeamName": "Outdial Queue-1"
},
"callFlowParams": {
    "OutdialQueue": {
        "description": "(vteam, The Outdial Queue.)",
        "name": "OutdialQueue",
        "qualifier": "vteam",
        "value": "3264",
        "valueDataType": "string"
    }
},
"callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "3264",
    "dnis": "8895579172",
    "isPaused": "false",
    "outdialTransferToQueueEnabled": "false",
    "pauseDuration": "10",
    "pauseResumeEnabled": "true",
    "recordInProgress": "true",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "133",
    "virtualTeamName": "Outdial Queue-1",
    "vteamId": "AXCLfZZU9S1oTdqE1OFZ"
},
"contactDirection": {
    "type": "OUTBOUND"
},
"currentVTeam": "3264",
"interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
"isFcManaged": false,
"isTerminated": false,
"media": {
    "1dd08726-bca5-4ce8-9b51-b20dca8ab154": {
        "holdTimestamp": 1612355834441,
        "isHold": true,
        "mType": "mainCall",
        "mediaMgr": "vmm",
        "mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
        "mediaType": "telephony",
        "participants": [
            ""
        ]
    }
},
"mediaChannel": "dialer",
"mediaType": "telephony",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"outboundType": "OUTDIAL",

```

```

"owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
"participants": {
  "": {
    "id": "",
    "pType": "Customer",
    "type": "Customer"
  },
  "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
    "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
    "consultState": null,
    "consultTimestamp": null,
    "dn": "8895579172",
    "hasJoined": true,
    "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "isConsulted": false,
    "isWrapUp": false,
    "joinTimestamp": 1612354635705,
    "lastUpdated": 1612354635705,
    "name": "John Doe",
    "pType": "Agent",
    "queueId": "3264",
    "queueMgrId": "aqm",
    "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
    "siteId": "472",
    "teamId": "960",
    "teamName": "Email_Team",
    "type": "Agent",
    "wrapUpTimestamp": null
  }
},
"previousVTeams": [
],
"state": "connected",
"workflowManager": null
},
"interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
"mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"trackingId": "3768264d-8588-465f-9ee6-0c25418ea9f7",
"type": "AgentContactHeld"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_8e8cabb7-6d5a-460d-9b8b-ecfad96448b3",
"type": "RoutingMessage"
}

```

unHold()

Resumes a voice call that is placed on hold.

Example

```

await Desktop.agentContact.unHold({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3",
  data: {
    mediaResourceId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
  }
});

// unHold Payload Type

```

```
{
  interactionId: string; data: {
    mediaResourceId: string
  }
}
```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options to resume the voice call that is placed on hold.	Yes
-->mediaResourceId	String	Unique identifier of the media resource.	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `AgentContactUnheld`; else, `AgentContactUnHoldFailed`.

Example Response

```
const unHoldResponse = {
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "destAgentId": null,
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "8895579172"
        },
        "ronaTimeout": {
          "agentEditable": false,
          "displayName": "ronaTimeout",
          "name": "ronaTimeout",
          "type": "STRING",
          "value": "30"
        },
        "virtualTeamName": {
          "agentEditable": false,
          "displayName": "virtualTeamName",
          "name": "virtualTeamName",
          "type": "STRING",
          "value": "Outdial Queue-1"
        }
      },
      "callAssociatedDetails": {
        "dn": "8895579172",
        "ronaTimeout": "30",
        "virtualTeamName": "Outdial Queue-1"
      }
    }
  }
}
```



```

"callFlowParams": {
  "OutdialQueue": {
    "description": "(vteam, The Outdial Queue.)",
    "name": "OutdialQueue",
    "qualifier": "vteam",
    "value": "3264",
    "valueDataType": "string"
  }
},
"callProcessingDetails": {
  "QMgrName": "aqm",
  "QueueId": "3264",
  "dnis": "8895579172",
  "isPaused": "false",
  "outdialTransferToQueueEnabled": "false",
  "pauseDuration": "10",
  "pauseResumeEnabled": "true",
  "recordInProgress": "true",
  "ronaTimeout": "30",
  "taskToBeSelfServiced": "false",
  "tenantId": "133",
  "virtualTeamName": "Outdial Queue-1",
  "vteamId": "AXCLfZZU9S1oTdQE1OFZ"
},
"contactDirection": {
  "type": "OUTBOUND"
},
"currentVTeam": "3264",
"interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
"isFcManaged": false,
"isTerminated": false,
"media": {
  "1dd08726-bca5-4ce8-9b51-b20dca8ab154": {
    "holdTimestamp": null,
    "isHold": false,
    "mType": "mainCall",
    "mediaMgr": "vmm",
    "mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
    "mediaType": "telephony",
    "participants": [
      ""
    ]
  }
},
"mediaChannel": "dialer",
"mediaType": "telephony",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"outboundType": "OUTDIAL",
"owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
"participants": {
  "": {
    "id": "",
    "pType": "Customer",
    "type": "Customer"
  },
  "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
    "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
    "consultState": null,
    "consultTimestamp": null,
    "dn": "8895579172",
    "hasJoined": true,
    "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "isConsulted": false,
    "isWrapUp": false,

```

```

        "joinTimestamp": 1612354635705,
        "lastUpdated": 1612354635705,
        "name": "John Doe",
        "pType": "Agent",
        "queueId": "3264",
        "queueMgrId": "aqm",
        "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
        "siteId": "472",
        "teamId": "960",
        "teamName": "Email_Team",
        "type": "Agent",
        "wrapUpTimestamp": null
      }
    },
    "previousVTeams": [

    ],
    "state": "connected",
    "workflowManager": null
  },
  "interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
  "mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "a5152cc8-004a-4e4c-b5d8-c9fc947f1e57",
  "type": "AgentContactUnheld"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_85f4d943-8e95-49c9-92c4-66047d526506",
"type": "RoutingMessage"
}

```

pauseRecording()

Pauses a call recording while obtaining sensitive information such as credit card information from the customer.

When configured by the administrator, voice calls are recorded for various reasons. When an agent handles sensitive customer information, the agent might want to pause the recording. The Pause/Resume functionality is available on the Agent Desktop through the Interaction Control pane. You can access the Pause/Resume functionality from your widget also.

Example

```

await Desktop.agentContact.pauseRecording({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3"
});
// Pausing a recording for telephony task

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes

Returns

{Object} The object with the retrieved data.



Note The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `ContactRecordingPaused`; else, `ContactRecordingPauseFailed`.

Example Response

```
const pauseRecordingResponse = {
  "data": {
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "8895579172"
        },
        "ronaTimeout": {
          "agentEditable": false,
          "displayName": "ronaTimeout",
          "name": "ronaTimeout",
          "type": "STRING",
          "value": "30"
        },
        "virtualTeamName": {
          "agentEditable": false,
          "displayName": "virtualTeamName",
          "name": "virtualTeamName",
          "type": "STRING",
          "value": "Outdial Queue-1"
        }
      },
      "callAssociatedDetails": {
        "dn": "8895579172",
        "ronaTimeout": "30",
        "virtualTeamName": "Outdial Queue-1"
      },
      "callFlowParams": {
        "OutdialQueue": {
          "description": "(vteam, The Outdial Queue.)",
          "name": "OutdialQueue",
          "qualifier": "vteam",
          "value": "3264",
          "valueDataType": "string"
        }
      },
      "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3264",
        "dnis": "8895579172",
        "isPaused": "true",
        "outdialTransferToQueueEnabled": "false",
        "pauseDuration": "10",
        "pauseResumeEnabled": "true",
        "recordInProgress": "true",
        "ronaTimeout": "30",
        "taskToBeSelfServiced": "false",
        "tenantId": "133",
        "virtualTeamName": "Outdial Queue-1",
        "vteamId": "AXCLfZZU9S1oTdqE1OFZ"
      }
    }
  },
}
```

■ pauseRecording()

```

    "contactDirection": {
      "type": "OUTBOUND"
    },
    "currentVTeam": "3264",
    "interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
    "isFcManaged": false,
    "isTerminated": false,
    "media": {
      "1dd08726-bca5-4ce8-9b51-b20dca8ab154": {
        "holdTimestamp": null,
        "isHold": false,
        "mType": "mainCall",
        "mediaMgr": "vmm",
        "mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
        "mediaType": "telephony",
        "participants": [
          ""
        ]
      }
    },
    "mediaChannel": "dialer",
    "mediaType": "telephony",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": "OUTDIAL",
    "owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "participants": {
      "": {
        "id": "",
        "pType": "Customer",
        "type": "Customer"
      },
      "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
        "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
        "consultState": null,
        "consultTimestamp": null,
        "dn": "8895579172",
        "hasJoined": true,
        "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
        "isConsulted": false,
        "isWrapUp": false,
        "joinTimestamp": 1612354635705,
        "lastUpdated": 1612354635705,
        "name": "John Doe",
        "pType": "Agent",
        "queueId": "3264",
        "queueMgrId": "aqm",
        "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
        "siteId": "472",
        "teamId": "960",
        "teamName": "Email_Team",
        "type": "Agent",
        "wrapUpTimestamp": null
      }
    },
    "previousVTeams": [
    ],
    "state": "connected",
    "workflowManager": null
  },
  "interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "b031f6d7-63a3-49d9-b377-e848bce41bd3",

```

```

        "type": "ContactRecordingPaused"
    },
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "trackingId": "notifs_52a00992-95e8-438c-ac44-8302da18d252",
    "type": "RoutingMessage"
}

```

resumeRecording()

Resumes a voice call recording that is paused.

Example

```

await Desktop.agentContact.resumeRecording({
  interactionId: "c837e6f7 - 699 a - 4736 - bb82 - 03 a6d58bb7f3",
  data: {
    autoResumed: true
  }
});
// Resuming recording for telephony task
// resumeRecording Payload Type

{
  interactionId: string; data: {
    autoResumed: boolean
  }
}

```

The following table lists the payload details:

Name	Type	Description	Required
interactionId	String	Unique identifier of the user interaction.	Yes
data	Object	Options to resume a voice call recording that is paused.	Yes
→autoResumed	Boolean	Determines whether the paused call must be manually resumed. <ul style="list-style-type: none"> • True—Voice call to be resumed automatically after the specified time is lapsed. • False—User to take action on the paused voice call based on the button defined. Example, Start Recording. 	Yes

Returns

{Object} The object with the retrieved data.



Note

The `type` parameter value in the response payload depends on the success or failure of the method. If the method is successful, the value is `ContactRecordingResumed`; else, `ContactRecordingResumeFailed`.

Example Response

```

const resumeRecordingResponse = {
  "data": {
    "autoResumed": true,
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "8895579172"
        },
        "ronaTimeout": {
          "agentEditable": false,
          "displayName": "ronaTimeout",
          "name": "ronaTimeout",
          "type": "STRING",
          "value": "30"
        },
        "virtualTeamName": {
          "agentEditable": false,
          "displayName": "virtualTeamName",
          "name": "virtualTeamName",
          "type": "STRING",
          "value": "Outdial Queue-1"
        }
      },
      "callAssociatedDetails": {
        "dn": "8895579172",
        "ronaTimeout": "30",
        "virtualTeamName": "Outdial Queue-1"
      },
      "callFlowParams": {
        "OutdialQueue": {
          "description": "(vteam, The Outdial Queue.)",
          "name": "OutdialQueue",
          "qualifier": "vteam",
          "value": "3264",
          "valueDataType": "string"
        }
      },
      "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3264",
        "dnis": "8895579172",
        "isPaused": "false",
        "outdialTransferToQueueEnabled": "false",
        "pauseDuration": "10",
        "pauseResumeEnabled": "true",
        "recordInProgress": "true",
        "ronaTimeout": "30",
        "taskToBeSelfServiced": "false",
        "tenantId": "133",
        "virtualTeamName": "Outdial Queue-1",
        "vteamId": "AXCLfZZU9S1oTdqE1OFZ"
      },
      "contactDirection": {
        "type": "OUTBOUND"
      },
      "currentVTeam": "3264",
      "interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
      "isFcManaged": false,
      "isTerminated": false,

```

```

"media": {
  "1dd08726-bca5-4ce8-9b51-b20dca8ab154": {
    "holdTimestamp": null,
    "isHold": false,
    "mType": "mainCall",
    "mediaMgr": "vmm",
    "mediaResourceId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
    "mediaType": "telephony",
    "participants": [
      ""
    ]
  }
},
"mediaChannel": "dialer",
"mediaType": "telephony",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"outboundType": "OUTDIAL",
"owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
"participants": {
  "": {
    "id": "",
    "pType": "Customer",
    "type": "Customer"
  },
  "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
    "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
    "consultState": null,
    "consultTimestamp": null,
    "dn": "8895579172",
    "hasJoined": true,
    "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "isConsulted": false,
    "isWrapUp": false,
    "joinTimestamp": 1612354635705,
    "lastUpdated": 1612354635705,
    "name": "John Doe",
    "pType": "Agent",
    "queueId": "3264",
    "queueMgrId": "aqm",
    "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
    "siteId": "472",
    "teamId": "960",
    "teamName": "Email_Team",
    "type": "Agent",
    "wrapUpTimestamp": null
  }
},
"previousVTeams": [
],
"state": "connected",
"workflowManager": null
},
"interactionId": "1dd08726-bca5-4ce8-9b51-b20dca8ab154",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"trackingId": "9cf5bfc4-d120-4330-9996-039a9937c52c",
"type": "ContactRecordingResumed"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_c04510a0-140a-439c-9ab4-68b6fdf6ea2e",
"type": "RoutingMessage"
}

```

Asynchronous Events

In order to subscribe to asynchronous events, use the following events:

- | | |
|-----------------------------|------------------------------------|
| • eAgentContact | • eAgentblindTransferred |
| • eAgentContactAssigned | • eAgentvteamTransfer |
| • eAgentContactAssignFailed | • eAgentConsultCreated |
| • eAgentContactWrappedUp | • eAgentConsultConferenced |
| • eAgentOfferContact | • eAgentConsultEnded |
| • eAgentOfferContactRona | • eAgentCtqCancelled |
| • eAgentOfferConsult | • eAgentConsultConferenceEnded |
| • eAgentWrapup | • eAgentConsulting |
| • eAgentContactHeld | • eAgentConsultFailed |
| • eAgentContactUnHeld | • eAgentConsultEndFailed |
| • eCallRecordingStarted | • eAgentCtqFailed |
| • eResumeRecording | • eAgentCtqCancelFailed |
| • ePauseRecording | • eAgentConsultConferenceEndFailed |
| • eConsultTransfer | |

Example

```
await Desktop.agentStateInfo.stateChange({
  state: "Available",
  auxCodeIdArray: "0",
});

// Get success/failure event here
Desktop.agentStateInfo.addEventListener("updated", updatedList =>
  logger.info(updatedList)
);
```

To remove an event subscription in order to avoid memory leaks, use the following options:

```
// Module supports removing added listeners like:
const listener = msg => logger.info(msg);
Desktop.agentContact.addEventListener("eAgentContact", listener);
Desktop.agentContact.removeEventListener("eAgentContact", listener);

// Module supports one-time added listeners like:
Desktop.agentContact.addOnceEventListener("eAgentContact", listener);
Desktop.agentContact.removeOnceEventListener("eAgentContact", listener);

// Module supports removing all listeners like:
Desktop.agentContact.removeAllEventListeners();
```




Note It is recommended to rely on the events, than on the responses of the asynchronous events such as stateChange.

Methods

addEventListener(event, handler)

Adds an event listener. This method listens to a particular event if generated. The event name can be specified as a string to the parameter.

Example

```
Desktop.agentContact.addEventListener("eAgentContact", (msg: Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Parameters

Name	Type	Description	Required
event	String	Type of an event.	Yes
handler	Function	The function that is invoked when the conditions are met.	Yes

removeEventListener(event, handler)

Removes an added event listener that has been attached with the [addEventListener\(event, handler\)](#) method.

Example

```
Desktop.agentContact.removeEventListener("eAgentContact", (msg: Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Parameters

Name	Type	Description	Required
event	String	Type of an event.	Yes
handler	Function	The function that is invoked when the conditions are met.	Yes

addOnceEventListener(event, handler)

Adds a one-time event listener.

Example

```
Desktop.agentContact.addOnceEventListener("eAgentContact", (msg: Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Parameters

Name	Type	Description	Required
event	String	Type of an event.	Yes

removeOnceEventListener(event, handler)

Name	Type	Description	Required
handler	Function	The function that is invoked when the conditions are met.	Yes

removeOnceEventListener(event, handler)

Removes a one-time added event listener.

Example

```
Desktop.agentContact.removeOnceEventListener("eAgentContact", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Parameters

Name	Type	Description	Required
event	String	Type of an event.	Yes
handler	Function	The function that is invoked when the conditions are met.	Yes

removeAllEventListeners(event, handler)

Removes all added event listeners.

Example

```
Desktop.agentContact.removeAllEventListeners();
```

Events

eAgentContact

Loads the desktop with the agent tasks.

Example

```
Desktop.agentContact.addEventListener("eAgentContact", (msg: Service.Aqm.Contact.AgentContact)
=> logger.info(msg));
```

Example Response

```
const eAgentContact = {
  type: 'RoutingMessage',
  orgId: 'f111e3af-1a45-42ef-9erf-4562354b8a25',
  trackingId: 'notifs_6f0ee13d-bb2f-4600-8631-c7d86fb08dfe',
  data: {
    mediaResourceId: '80c213d5-2747-4bf4-8a0c-9c45b0afdf2a',
    eventType: 'RoutingMessage',
    agentId: '9738292f-3a68-4909-bdea-b541cccd6935',
    trackingId: '68bca42c-8a50-4cf1-8c18-f5229b04afa9',
    interaction: {
      isFcManaged: false,
      isTerminated: false,
      mediaType: 'telephony',
      previousVTeams: ['AXCZ0YCAxrf9I0XFBI7b'],
      state: 'new',
      currentVTeam: '3268',
      participants: {
```

```

'+19997770095': {
  id: '+19997770095',
  pType: 'Customer',
  type: 'Customer'
},
'9738292f-3a68-4909-bdea-b541cccd6935': {
  joinTimestamp: null,
  name: '',
  pType: 'Agent',
  teamName: 'Email_Team',
  lastUpdated: 1593440068730,
  teamId: '960',
  isConsulted: false,
  hasJoined: false,
  dn: '9997770093',
  queueId: '3268',
  id: '9738292f-3a68-4909-bdea-b541cccd6935',
  sessionId: '054ebc22-b34e-4e1b-8730-da294093f813',
  queueMgrId: 'aqm',
  siteId: '472',
  type: 'Agent',
  channelId: '53e05ff1-69c7-425f-a660-5f9e21f0a4bb',
  isWrapUp: false
}
},
interactionId: '80c213d5-2747-4bf4-8a0c-9c45b0afdf2a',
orgId: 'f111e3af-1a45-42ef-9erf-4562354b8a25',
callProcessingDetails: {
  QMgrName: 'aqm',
  taskToBeSelfServiced: 'false',
  ani: '+19997770095',
  dnis: '+12147659000',
  tenantId: '133',
  QueueId: '3268',
  vteamId: '3268',
  jscriptId: 'AXCZ4c3mjkgwAuS7vSIU',
  virtualTeamName: 'Queue - Telephony',
  customerName: 'Jane Doe',
  ronaTimeout: '30',
  reasonCode: 'Credit',
  IvrPath: 'EOI',
  pathId: 'StartCall PlayDone out out out'
},
media: {
  '80c213d5-2747-4bf4-8a0c-9c45b0afdf2a': {
    mediaResourceId: '80c213d5-2747-4bf4-8a0c-9c45b0afdf2a',
    mediaType: 'telephony',
    mediaMgr: 'vmm',
    participants: ['+19997770095'],
    mType: 'mainCall',
    isHold: false
  }
},
owner: '9738292f-3a68-4909-bdea-b541cccd6935',
mediaChannel: 'broadcloud',
contactDirection: {
  type: 'INBOUND'
},
callFlowParams: {
  Play2: {
    name: 'Play2',
    qualifier: '',
    description: '(A valid text.)',
    valueDataType: 'string',

```

```

        value: 'Welcome to Contact Center'
      },
      Queue3: {
        name: 'Queue3',
        qualifier: 'vteam',
        description: '(vteam, A valid VTeam.)',
        valueDataType: 'string',
        value: '3268'
      }
    },
    interactionId: '80c213d5-2747-4bf4-8a0c-9c45b0afdf2a',
    orgId: 'f111e3af-1a45-42ef-9erf-4562354b8a25',
    queueMgr: 'aqm',
    type: 'AgentContact',
    isConsulted: false,
    consultMediaResourceId: null
  }
};

```

eAgentContactAssigned

New task or contact has been assigned successfully.

Example

```
Desktop.agentContact.addEventListener("eAgentContactAssigned", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [accept\(\)](#).

eAgentContactAssignFailed

Assigned task or contact did not succeed.

Example

```
Desktop.agentContact.addEventListener("eAgentContactAssignFailed", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [accept\(\)](#).

eAgentContactWrappedUp

Accepted task or contact has been wrapped up.

Example

```
Desktop.agentContact.addEventListener("eAgentContactWrappedUp", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [wrapup\(\)](#).

eAgentOfferContact

New task or contact has been routed to an agent.

Example

```
Desktop.agentContact.addEventListener("eAgentOfferContact", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Example Response

```

const eAgentOfferContact = {
  "data": {
    "agentId": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "ani": {
          "value": "janedoe@gmail.com"
        },
        "category": {
          "value": "Automation"
        },
        "customerName": {
          "value": "Jane Doe"
        },
        "dn": {
          "value": "Desktop"
        },
        "entryPointId": {
          "value": "AXCqFyz1G2pKap9PI-mW"
        },
        "guestId": {
          "value":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS83ZjA3YjJmMC1jNTMyLTQ5MDktYTktZmNi0yYTA3MTVmZGIwMTA"
        },
        "mediaChannel": {
          "value": "web"
        },
        "reason": {
          "value": "Test"
        },
        "reasonCode": {
          "value": "Automation"
        },
        "ronaTimeout": {
          "value": "30"
        },
        "roomTitle": {
          "value": "Help Jane Doe with Automation"
        },
        "taskToBeSelfServiced": {
          "value": "false"
        },
        "templateId": {
          "value": "b57990c0-5ec6-11ea-96ee-5df5aef56329"
        },
        "templateName": {
          "value": "Desktop"
        },
        "virtualTeamName": {
          "value": "Chat_Queue"
        }
      },
      "callAssociatedDetails": {
        "ani": "janedoe@gmail.com",
        "category": "Automation",
        "customerName": "Jane Doe",
        "dn": "Desktop",
        "entryPointId": "AXCqFyz1G2pKap9PI-mW",
        "guestId":
"Y21zY29zcGFyazovL3VzL1BFTlBMRS83ZjA3YjJmMC1jNTMyLTQ5MDktYTktZmNi0yYTA3MTVmZGIwMTA",
        "mediaChannel": "web",
        "reason": "Test",
        "reasonCode": "Automation",

```

```

        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Automation",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "virtualTeamName": "Chat_Queue"
    },
    "callFlowParams": {
        "Automation": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Automation",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Debit": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Debit",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        },
        "Sales": {
            "description": "(vteam, A valid VTeam.)",
            "name": "Sales",
            "qualifier": "vteam",
            "value": "3270",
            "valueDataType": "string"
        }
    },
    "callProcessingDetails": {
        "QMgrName": "aqm",
        "QueueId": "3270",
        "ani": "janedoe@gmail.com",
        "category": "Automation",
        "customerName": "Jane Doe",
        "dnis": "Desktop",
        "entryPointId": "AXCqFyzlG2pKap9PI-mW",
        "guestId":
"Y2lzMzY2ZzcGFyYXovL3VzL1BFTlBMRs83ZjA3YjJmMC1jNTMyLTQ5MDktYTtkZni0yYTA3MTVmZGIwMTA",
        "mediaChannel": "web",
        "reason": "Test",
        "reasonCode": "Automation",
        "ronaTimeout": "30",
        "roomTitle": "Help Jane Doe with Automation",
        "taskToBeSelfServiced": "false",
        "templateId": "b57990c0-5ec6-11ea-96ee-5df5aef56329",
        "templateName": "Desktop",
        "tenantId": "133",
        "virtualTeamName": "Chat_Queue",
        "vteamId": "AXCqFyzlG2pKap9PI-mW"
    },
    "contactDirection": {
        "type": "INBOUND"
    },
    "currentVTeam": "3270",
    "interactionId": "8d3a56fa-4172-11eb-abaf-1b20df734401",
    "isFcManaged": false,
    "isTerminated": false,
    "media": {
"Y2lzMzY2ZzcGFyYXovL3VzL1JPT00vOTAxMmEzZjAtNDE3Mi0xMWVlLWE1N2QtZWY2N2MwZTM5YmFh": {
        "holdTimestamp": null,
        "isHold": false,

```

```

        "mType": "mainCall",
        "mediaMgr": "cmm",
        "mediaResourceId":
"Y21zY29zcGFyazovL3VzL1JPT00vOTAxMmEzZjAtNDE3Mi0xMWViLWE1N2QtZWY2N2MwZTMmYmFh",
        "mediaType": "chat",
        "participants": ["janedoe@gmail.com"]
    }
},
"mediaChannel": "web",
"mediaType": "chat",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"outboundType": null,
"owner": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
"participants": {
    "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed": {
        "channelId": "bff9350a-9d2a-489a-a8d5-d9dacea4b692",
        "consultState": null,
        "consultTimestamp": null,
        "dn": "9997770110",
        "hasJoined": false,
        "id": "7d12d9ea-e8e0-41ee-81bf-c11a685b64ed",
        "isConsulted": false,
        "isWrapUp": false,
        "joinTimestamp": null,
        "lastUpdated": 1608324587206,
        "name": "uui-agent4 uui-agent4",
        "pType": "Agent",
        "queueId": "3270",
        "queueMgrId": "aqm",
        "sessionId": "102d2096-bcc0-45bb-a583-a02f6c188071",
        "siteId": "473",
        "teamId": "1940",
        "teamName": "Medical Help",
        "type": "Agent",
        "wrapUpTimestamp": null
    },
    "janedoe@gmail.com": {
        "id": "janedoe@gmail.com",
        "pType": "Customer",
        "type": "Customer"
    }
},
"previousVTeams": [],
"state": "new"
},
"interactionId": "8d3a56fa-4172-11eb-abaf-1b20df734401",
"mediaResourceId":
"Y21zY29zcGFyazovL3VzL1JPT00vOTAxMmEzZjAtNDE3Mi0xMWViLWE1N2QtZWY2N2MwZTMmYmFh",
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"queueMgr": "aqm",
"ronaTimeout": 30,
"trackingId": "429bb73d-7463-4661-94aa-a3230d60e88a",
"type": "AgentOfferContact"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_f23e73e4-b130-4bce-b110-1c1da7225ffa",
"type": "RoutingMessage"
}

```

eAgentOfferContactRona

Agent failed to accept the task or contact request within the maximum available time. The system changes the agent state from **Available** to **RONA**.

Example

```
Desktop.agentContact.addEventListener("eAgentOfferContactRona", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [decline\(\)](#).

eAgentOfferConsult

Accepted task or contact consult request has been routed to an agent.

Example

```
Desktop.agentContact.addEventListener("eAgentOfferConsult", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Example Response

```
export const eAgentOfferConsult = {
  type: 'RoutingMessage',
  orgId: 'f111e3af-1a45-42ef-9erf-4562354b8a25',
  trackingId: 'notifs_50e9d868-71c0-48f2-9880-4a12fc505988',
  data: {
    consultMediaResourceId:
'consult_Y2lzY29zcGFyazovL3VzL1JPT00vODZkMGQ2MDAOTQ0YyOxMwVhLWE4MjMtNzFjMjA3YWRLYjAx',
    eventType: 'RoutingMessage',
    agentId: '8546f083-74ae-4510-b2a6-c8e21f01689b',
    consultingAgentId: '9738292f-3a68-4909-bdea-b541cccd6935',
    trackingId: 'ca074768-9d2b-427e-86af-3745c2a8cc60',
    interaction: {
      isFcManaged: false,
      isTerminated: false,
      mediaType: 'chat',
      previousVTeams: [],
      state: 'consult',
      currentVTeam: '3270',
      participants: {
        'mohaksin@cisco.com': {
          id: 'mohaksin@cisco.com',
          pType: 'Customer',
          type: 'Customer'
        },
        '8546f083-74ae-4510-b2a6-c8e21f01689b': {
          name: '',
          pType: 'Agent',
          teamName: 'Dont_Use_Team',
          joinTimestamp: null,
          lastUpdated: 1589287209385,
          teamId: '964',
          isConsulted: true,
          hasJoined: false,
          dn: '9997770111',
          queueId: '3270',
          id: '8546f083-74ae-4510-b2a6-c8e21f01689b',
          sessionId: 'dead6d11-3d8f-42ea-87d9-b94d5f5b6370',
          queueMgrId: 'aqm',
          siteId: '472',
          type: 'Agent',
          channelId: 'a4297667-6d16-4d9d-ad2f-b21fee6b97fe',
          isWrapUp: false
        },
        '9738292f-3a68-4909-bdea-b541cccd6935': {
          name: 'vivek',
          pType: 'Agent',
          teamName: 'Dont_Use_Team',
```



```

        joinTimestamp: 1594295035415,
        lastUpdated: 1589287209385,
        teamId: '964',
        isConsulted: false,
        hasJoined: true,
        dn: '9997770111',
        queueId: '3270',
        id: '9738292f-3a68-4909-bdea-b541cccd6935',
        sessionId: 'dead6d11-3d8f-42ea-87d9-b94d5f5b6370',
        queueMgrId: 'aqm',
        siteId: '472',
        type: 'Agent',
        channelId: 'a4297667-6d16-4d9d-ad2f-b21fee6b97fe',
        isWrapUp: false
    },
    },
    interactionId: 'cd4f3721-944b-11ea-a5e5-bdac2bb8b026',
    orgId: 'f111e3af-1a45-42ef-9erf-4562354b8a25',
    callProcessingDetails: {
        QMgrName: 'aqm',
        taskToBeSelfServiced: 'false',
        ani: 'mohaksin@cisco.com',
        dnis: 'AgentX',
        tenantId: '133',
        QueueId: '3270',
        vteamId: 'AXCqFyz1G2pKap9PI-mW',
        virtualTeamName: 'Chat_Queue',
        customerName: 'Mohak',
        ronaTimeout: '30',
        reasonCode: 'Sales',
        reason: 'Lockdown challeeya',
        category: 'Sales'
    },
    media: {
        Y21zY29zcGFyazovL3VzL1JPT00vODZkMGQ2MDAtOTQ0Yy0xMWVhLWE4MjMtNzFjMjA3YWRLYjAx:
    {
        mediaResourceId:
        'Y21zY29zcGFyazovL3VzL1JPT00vODZkMGQ2MDAtOTQ0Yy0xMWVhLWE4MjMtNzFjMjA3YWRLYjAx',
        mediaType: 'chat',
        mediaMgr: 'cmm',
        participants: ['mohaksin@cisco.com'],
        mType: 'mainCall',
        isHold: false
    }
    },
    owner: '8546f083-74ae-4510-b2a6-c8e21f01689b',
    mediaChannel: 'web',
    contactDirection: {
        type: 'INBOUND'
    },
    },
    callFlowParams: {
        Sales: {
            name: 'Sales',
            qualifier: 'vteam',
            description: '(vteam, A valid VTeam.)',
            valueDataType: 'string',
            value: '3270'
        },
        Debit: {
            name: 'Debit',
            qualifier: 'vteam',
            description: '(vteam, A valid VTeam.)',
            valueDataType: 'string',
            value: '3270'
        }
    }
}

```

```

    },
    Automation: {
      name: 'Automation',
      qualifier: 'vteam',
      description: '(vteam, A valid VTeam.)',
      valueDataType: 'string',
      value: '3270'
    }
  },
  ronaTimeout: 30,
  interactionId: 'cd4f3721-944b-11ea-a5e5-bdac2bb8b026',
  orgId: 'f111e3af-1a45-42ef-9erf-4562354b8a25',
  queueMgr: 'aqm',
  type: 'AgentOfferConsult'
}
};

```

eAgentWrapup

Accepted task or contact has been moved to a wrap up state.

Example

```
Desktop.agentContact.addEventListener("eAgentWrapup", (msg: Service.Aqm.Contact.AgentContact)
=> logger.info(msg));
```

For more information on example response, see [end\(\)](#).

eAgentContactHeld

Accepted task or contact has been put on hold.

Example

```
Desktop.agentContact.addEventListener("eAgentContactHeld", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [hold\(\)](#).

eAgentContactUnHeld

Accepted task or contact has been resumed.

Example

```
Desktop.agentContact.addEventListener("eAgentContactUnHeld", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [unHold\(\)](#).

eCallRecordingStarted

Call recording for an accepted task or contact has started.

Example

```
Desktop.agentContact.addEventListener("eCallRecordingStarted", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

Example Response

```
const eCallRecordingStarted = {
  "data": {
```

```

"eventType": "RoutingMessage",
"interaction": {
  "callAssociatedData": {
    "ani": {
      "agentEditable": false,
      "displayName": "ani",
      "name": "ani",
      "type": "STRING",
      "value": "9997652131"
    },
    "dn": {
      "agentEditable": false,
      "displayName": "dn",
      "name": "dn",
      "type": "STRING",
      "value": "9997651073"
    },
    "ronaTimeout": {
      "agentEditable": false,
      "displayName": "ronaTimeout",
      "name": "ronaTimeout",
      "type": "STRING",
      "value": "30"
    },
    "virtualTeamName": {
      "agentEditable": false,
      "displayName": "virtualTeamName",
      "name": "virtualTeamName",
      "type": "STRING",
      "value": "qaus1-gt-requeue"
    }
  },
  "callAssociatedDetails": {
    "ani": "*****",
    "dn": "*****",
    "ronaTimeout": "30",
    "virtualTeamName": "qaus1-gt-requeue"
  },
  "callFlowParams": {
    "OutdialQueue": {
      "description": "(vteam, The Outdial Queue.)",
      "name": "OutdialQueue",
      "qualifier": "vteam",
      "value": "4125",
      "valueDataType": "string"
    }
  },
  "callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "4125",
    "agent_ani": "9997652131",
    "ani": "*****",
    "dnis": "*****",
    "doNotRouteToAgents": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
    "outdialParkEpId": "AXRnbLmz1OI4n5klYr5E",
    "outdialTransferToQueueEnabled": "true",
    "pauseDuration": "30",
    "pauseResumeEnabled": "true",
    "recordInProgress": "true",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "82",
    "virtualTeamName": "qaus1-gt-requeue",
    "vteamId": "6838",

```

```

    "vteamType": "inboundqueue"
  },
  "contactDirection": {
    "type": "OUTBOUND"
  },
  "currentVTeam": "6838",
  "interactionId": "28a8f08f-0d2f-46f9-9751-a2d9536886c1",
  "isFcManaged": false,
  "isTerminated": false,
  "media": {
    "28a8f08f-0d2f-46f9-9751-a2d9536886c1": {
      "holdTimestamp": null,
      "isHold": false,
      "mType": "mainCall",
      "mediaMgr": "vmm",
      "mediaResourceId": "28a8f08f-0d2f-46f9-9751-a2d9536886c1",
      "mediaType": "telephony",
      "participants": [
        "*****",
        "*****"
      ]
    }
  },
  "mediaChannel": "dialer",
  "mediaType": "telephony",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "outboundType": "OUTDIAL",
  "owner": "5b24107f-aae2-45e6-9bf8-a36d18b6f0e5",
  "participants": {
    "5b24107f-aae2-45e6-9bf8-a36d18b6f0e5": {
      "channelId": "1b61e3f4-9c33-4241-9304-9ea054b88be0",
      "consultState": null,
      "consultTimestamp": null,
      "dn": "*****",
      "hasJoined": true,
      "id": "5b24107f-aae2-45e6-9bf8-a36d18b6f0e5",
      "isConsulted": false,
      "isWrapUp": false,
      "joinTimestamp": 1612429637332,
      "lastUpdated": 1612429637332,
      "name": "qaus1-gt-requeue-user qaus1-gt-requeue-user",
      "pType": "Agent",
      "queueId": "6838",
      "queueMgrId": "aqm",
      "sessionId": "56d94e8d-fbd0-4933-936d-7c19f422ec47",
      "siteId": "205",
      "teamId": "1837",
      "teamName": "integ-test-team",
      "type": "Agent",
      "wrapUpTimestamp": null
    },
    "*****": {
      "id": "9997652131",
      "pType": "Customer",
      "type": "Customer"
    },
    "9b036d89-930c-4187-a5bd-5dbb1439fe41": {
      "channelId": "3de6706f-95b7-485a-9304-30928f7ee52e",
      "consultState": null,
      "consultTimestamp": null,
      "dn": "*****",
      "hasJoined": true,
      "id": "9b036d89-930c-4187-a5bd-5dbb1439fe41",
      "isConsulted": false,

```

```

        "isWrapUp": true,
        "joinTimestamp": 1612429625705,
        "lastUpdated": 1612429636839,
        "name": "qaus1-gt-user1 qaus1-gt-user1",
        "pType": "Agent",
        "queueId": "4125",
        "queueMgrId": "aqm",
        "sessionId": "90896c91-0f77-4bc6-aec3-488a34ca88d3",
        "siteId": "205",
        "teamId": "598",
        "teamName": "integ-test-team",
        "type": "Agent",
        "wrapUpTimestamp": 1612429636839
      }
    },
    "previousVTeams": [
      "4125"
    ],
    "state": "connected",
    "workflowManager": null
  },
  "interactionId": "28a8f08f-0d2f-46f9-9751-a2d9536886c1",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "trackingId": "795e3411-191d-4524-b01c-bf672126fe02",
  "type": "ContactRecordingStarted"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_9e18c618-cdd6-4bf7-be18-cafd250faabc",
"type": "RoutingMessage"
}

```

eResumeRecording

Resumes a voice call recording for an accepted task or contact.

Example

```
Desktop.agentContact.addEventListener("eResumeRecording", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [resumeRecording\(\)](#).

ePauseRecording

Pauses a call recording for an accepted task or contact.

Example

```
Desktop.agentContact.addEventListener("ePauseRecording", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [pauseRecording\(\)](#).

eConsultTransfer

Consult transfer request for an accepted task or contact.

Example

```
Desktop.agentContact.addEventListener("eConsultTransfer", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consultTransfer\(\)](#).

eAgentblindTransferred

Transfers an accepted task or contact to an agent or queue.

Example

```
Desktop.agentContact.addEventListener("eAgentblindTransferred", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [blindTransfer\(\)](#).

eAgentvteamTransfer

Transfers an accepted task or contact to a queue.

Example

```
Desktop.agentContact.addEventListener("eAgentvteamTransfer", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [vTeamTransfer\(\)](#).

eAgentConsultCreated

Consult request for an accepted task or contact has been created.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultCreated", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consult\(\)](#).

eAgentConsultConferenced

Consult conference request for an accepted task or contact has been connected.



Note

Event is received when conference is established for all channels except email.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultConferenced", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consultConference\(\)](#).

eAgentConsultEnded

Consult request for an accepted task or contact has ended.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultEnded", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consultEnd\(\)](#).

eAgentCtqCancelled

Consult request to the queue has been canceled.

Example

```
Desktop.agentContact.addEventListener("eAgentCtqCancelled", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [cancelCtq\(\)](#).

eAgentConsultConferenceEnded

Consult conference request for an accepted task or contact has ended.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultConferenceEnded", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consultEnd\(\)](#).

eAgentConsulting

Agent has started consulting an incoming task or contact.

Example

```
Desktop.agentContact.addEventListener("eAgentConsulting", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consultAccept\(\)](#).

eAgentConsultFailed

Consult request for an accepted task or contact did not succeed.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultFailed", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consult\(\)](#).

eAgentConsultEndFailed

Request to end consult for an accepted task or contact did not succeed.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultEndFailed", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consultEnd\(\)](#).

eAgentCtqFailed

Consult request to the queue did not succeed.

Example

```
Desktop.agentContact.addEventListener("eAgentCtqFailed", (msg:
Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [consult\(\)](#).

eAgentCtqCancelFailed

Request to cancel the consult to the queue did not succeed.

Example

```
Desktop.agentContact.addEventListener("eAgentCtqCancelFailed", (msg: Service.Aqm.Contact.AgentContact) => logger.info(msg));
```

For more information on example response, see [cancelCtq\(\)](#).

eAgentConsultConferenceEndFailed

Request to end the consult conference request did not succeed.

Example

```
Desktop.agentContact.addEventListener("eAgentConsultConferenceEndFailed", (msg: Service.Aqm.Contact.AgentContact) => logger.info(msg));
```




CHAPTER 10

Dialer Module

The `Desktop.dialer` module makes requests and listens to notification events related to the outdial activity (outgoing customer calls initiated by agents).

Example

```
import {
  Desktop
} from "@wxcc-desktop/sdk";

...

const outdial = await Desktop.dialer.startOutdial({
  data: {
    entryPointId: "1212312567",
    destination: "12078995678",
    direction: "OUTBOUND",
    attributes: {},
    mediaType: "telephony",
    outboundType: "CourtesyCallback"
  }
});

logger.info("Dialer outdial: ", outdial);

// List of available subscription events:
Desktop.dialer.addEventListener("eOutdialFailed", msg => logger.info(msg));
```

To remove an event subscription in order to avoid memory leaks, you have the following options:

```
// Module supports removing added listeners like:
const listener = msg => logger.info(msg);
Desktop.dialer.addEventListener("eOutdialFailed", listener);
Desktop.dialer.removeEventListener("eOutdialFailed", listener);

// Module supports one-time added listeners like:
Desktop.dialer.addOnceEventListener("eOutdialFailed", listener);
Desktop.dialer.removeOnceEventListener("eOutdialFailed", listener);

// Module supports removing all listeners like:
Desktop.dialer.removeAllEventListeners();
```

- [Methods, on page 122](#)
- [Events, on page 124](#)

Methods

startOutdial(data)

Initiates an outdial call.

Example

```
await Desktop.dialer.startOutdial({
  data: {
    entryPointId: "1212312567",
    destination: "12078995678",
    direction: "OUTBOUND",
    attributes: {},
    mediaType: "telephony",
    outboundType: "CourtesyCallback"
  }
});
```

Parameters

Name	Type	Description	Required
data	Object	Options for the outdial call.	Yes
-->entryPointId	String	Unique identifier of the entry point.	Yes
-->destination	String	Dial number of the end recipient.	Yes
-->direction	String	Direction of call from a dialer to recipient.	Yes
-->attributes	Object	Options of the attributes.	Yes
-->mediaType	String	The media channel type such as telephony, social, email, and chat.	Yes
-->outboundType	String	Type of an outdial call such as OUTDIAL, CourtesyCallback.	Yes

Returns

{Object} The object with the retrieved data.

Example Response

```
const startOutdialResponse = {
  "data": {
    "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "eventType": "RoutingMessage",
    "interaction": {
      "callAssociatedData": {
        "dn": {
          "agentEditable": false,
          "displayName": "dn",
          "name": "dn",
          "type": "STRING",
          "value": "8895579172"
        }
      }
    }
  }
}
```

```

    },
    "ronaTimeout": {
      "agentEditable": false,
      "displayName": "ronaTimeout",
      "name": "ronaTimeout",
      "type": "STRING",
      "value": "30"
    },
    "virtualTeamName": {
      "agentEditable": false,
      "displayName": "virtualTeamName",
      "name": "virtualTeamName",
      "type": "STRING",
      "value": "Outdial Queue-1"
    }
  },
  "callAssociatedDetails": {
    "dn": "8895579172",
    "ronaTimeout": "30",
    "virtualTeamName": "Outdial Queue-1"
  },
  "callFlowParams": {
    "OutdialQueue": {
      "description": "(vteam, The Outdial Queue.)",
      "name": "OutdialQueue",
      "qualifier": "vteam",
      "value": "3264",
      "valueDataType": "string"
    }
  },
  "callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "3264",
    "dnis": "8895579172",
    "outdialTransferToQueueEnabled": "false",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "133",
    "virtualTeamName": "Outdial Queue-1",
    "vteamId": "AXCLfZZU9S1oTdqE1OFZ"
  },
  "contactDirection": {
    "type": "OUTBOUND"
  },
  "currentVTeam": "3264",
  "interactionId": "8da312bb-5349-4d97-89d9-7bedd01b8782",
  "isFcManaged": false,
  "isTerminated": false,
  "media": {
    "8da312bb-5349-4d97-89d9-7bedd01b8782": {
      "holdTimestamp": null,
      "isHold": false,
      "mType": "mainCall",
      "mediaMgr": "vmm",
      "mediaResourceId": "8da312bb-5349-4d97-89d9-7bedd01b8782",
      "mediaType": "telephony",
      "participants": []
    }
  },
  "mediaChannel": "dialer",
  "mediaType": "telephony",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "outboundType": "OUTDIAL",
  "owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",

```

```

    "participants": {
      "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
        "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
        "consultState": null,
        "consultTimestamp": null,
        "dn": "8895579172",
        "hasJoined": false,
        "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
        "isConsulted": false,
        "isWrapUp": false,
        "joinTimestamp": null,
        "lastUpdated": 1612345959131,
        "name": "Jane Doe",
        "pType": "Agent",
        "queueId": "3264",
        "queueMgrId": "aqm",
        "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
        "siteId": "472",
        "teamId": "960",
        "teamName": "Email_Team",
        "type": "Agent",
        "wrapUpTimestamp": null
      }
    },
    "previousVTeams": [],
    "state": "new",
    "workflowManager": null
  },
  "interactionId": "8da312bb-5349-4d97-89d9-7bedd01b8782",
  "mediaResourceId": "8da312bb-5349-4d97-89d9-7bedd01b8782",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueMgr": "aqm",
  "ronaTimeout": 30,
  "trackingId": "cfe421df-f107-42f3-945b-a6da5dd13ccd",
  "type": "AgentOfferContact"
},
"orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
"trackingId": "notifs_a6cfbcc4-7a8c-4a2e-a9d5-6c23683cdcb7",
"type": "RoutingMessage"
}

```

Events

eOutdialFailed

Outdial call did not succeed.

Example

```
Desktop.dialer.addEventListener("eOutdialFailed", msg => logger.info(msg));
```

Example Response

```

const eOutdialFailed = {
  "agentId": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
  "eventType": "RoutingMessage",
  "interaction": {
    "callAssociatedData": {
      "dn": {
        "agentEditable": false,
        "displayName": "dn",

```

```

        "name": "dn",
        "type": "STRING",
        "value": "8895579172"
    },
    "ronaTimeout": {
        "agentEditable": false,
        "displayName": "ronaTimeout",
        "name": "ronaTimeout",
        "type": "STRING",
        "value": "30"
    },
    "virtualTeamName": {
        "agentEditable": false,
        "displayName": "virtualTeamName",
        "name": "virtualTeamName",
        "type": "STRING",
        "value": "Outdial Queue-1"
    }
},
"callAssociatedDetails": {
    "dn": "8895579172",
    "ronaTimeout": "30",
    "virtualTeamName": "Outdial Queue-1"
},
"callFlowParams": {
    "OutdialQueue": {
        "description": "(vteam, The Outdial Queue.)",
        "name": "OutdialQueue",
        "qualifier": "vteam",
        "value": "3264",
        "valueDataType": "string"
    }
},
"callProcessingDetails": {
    "QMgrName": "aqm",
    "QueueId": "3264",
    "dnis": "8895579172",
    "outdialTransferToQueueEnabled": "false",
    "ronaTimeout": "30",
    "taskToBeSelfServiced": "false",
    "tenantId": "133",
    "virtualTeamName": "Outdial Queue-1",
    "vteamId": "AXCLfZZU9S1oTdqE1OFZ"
},
"contactDirection": {
    "type": "OUTBOUND"
},
"currentVTeam": "3264",
"interactionId": "d87dac90-4f81-486c-8deb-b94bbcd21e92",
"isFcManaged": false,
"isTerminated": true,
"media": {
    "d87dac90-4f81-486c-8deb-b94bbcd21e92": {
        "holdTimestamp": null,
        "isHold": false,
        "mType": "mainCall",
        "mediaMgr": "vmm",
        "mediaResourceId": "d87dac90-4f81-486c-8deb-b94bbcd21e92",
        "mediaType": "telephony",
        "participants": [

        ]
    }
}
},

```

```

    "mediaChannel": "dialer",
    "mediaType": "telephony",
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
    "outboundType": "OUTDIAL",
    "owner": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
    "participants": {
      "7c867aa9-ec768-341a-b767-e5hd6ae7g701": {
        "channelId": "0717a2ce-76cd-4ba4-9053-71f2c78168b8",
        "consultState": null,
        "consultTimestamp": null,
        "dn": "8895579172",
        "hasJoined": false,
        "id": "7c867aa9-ec768-341a-b767-e5hd6ae7g701",
        "isConsulted": false,
        "isWrapUp": false,
        "joinTimestamp": null,
        "lastUpdated": 1612346591051,
        "name": "John Doe",
        "pType": "Agent",
        "queueId": "3264",
        "queueMgrId": "aqm",
        "sessionId": "3d017488-527a-4e89-9313-5d4eb353c789",
        "siteId": "472",
        "teamId": "960",
        "teamName": "Email_Team",
        "type": "Agent",
        "wrapUpTimestamp": null
      }
    },
    "previousVTeams": [
      ],
    "state": "new",
    "workflowManager": null
  },
  "interactionId": "d87dac90-4f81-486c-8deb-b94bbcd21e92",
  "mediaType": "telephony",
  "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",
  "queueId": "3264",
  "queueMgr": "aqm",
  "reason": "CUSTOMER_UNAVAILABLE",
  "reasonCode": 500,
  "trackingId": "09f0d9fd-6cad-48a4-a906-be1b09a8fdc0",
  "type": "AgentOutboundFailed"
}

```



CHAPTER 11

Screen Pop Module

The `Desktop.screenpop` module makes requests and listens to notification events related to the screen pop entity.

Example:

```
Desktop.screenpop.addEventListener("eScreenPop", msg => logger.info(msg));
```

To remove an event subscription in order to avoid memory leaks, you have the following options:

```
// Module supports removing added listeners like:
const listener = msg => logger.info(msg);
Desktop.screenpop.addEventListener("eScreenPop", listener);
Desktop.screenpop.removeEventListener("eScreenPop", listener);

// Module supports one-time added listeners like:
Desktop.screenpop.addOnceEventListener("eScreenPop", listener);
Desktop.screenpop.removeOnceEventListener("eScreenPop", listener);

// Module supports removing all listeners like:
Desktop.screenpop.removeAllEventListeners();
```

- [Events, on page 127](#)

Events

eScreenPop

Listens to an event that displays the screen pop.

Example

```
Desktop.screenpop.addEventListener("eScreenPop", listener);
```

Example Response

```
const eScreenPop = {
  "data": {
    "agentId": "1736827c-10ef-4b2a-90e0-7d2a3743ded3",
    "interactionId": "4745472e-cfa2-47f3-9fc1-16359516d648",
    "queryParameters": null,
    "screenPopUrl": "https://youtube.com",
    "target": "insideDesktop",
    "type": "ScreenPop"
  },
}
```

```
    "orgId": "f111e3af-1a45-42ef-9erf-4562354b8a25",  
    "trackingId": "notifs_90a4bf42-ba95-47a6-b8cb-cl1ed105ff5ea",  
    "type": "RoutingMessage"  
  }  
}
```




CHAPTER 12

Shortcut Key Module

The `Desktop.shortcutKey` module intends to register and call the shortcut key actions from widgets. You must know the list of existing shortcut keys in Agent Desktop to avoid conflicts. For more information, see the *Access Keyboard Shortcuts* section in the *Introduction* chapter of the [Cisco Webex Contact Center Agent Desktop User Guide](#).

Example:

```
import {
  Desktop
} from "@wxc-desktop/sdk";

console.log(Desktop.shortcutKey.DEFAULT_SHORTCUT_KEYS); //=> logs default shortcut keys

console.log(Desktop.shortcutKey.MODIFIERS); //=> logs keys modifiers

console.log(Desktop.shortcutKey.REGISTERED_KEYS); //=> logs registered keys without modifiers

console.log(Desktop.shortcutKey.getRegisteredKeys()); //=> logs service registered keys
with modifiers (full information)

Desktop.shortcutKey.listenKeyPress((event) => {
  ...
}); //=> listen shortcuts key press

Desktop.shortcutKey.listenKeyConflict((event) => {
  ...
}); //=> listen shortcuts key conflict

Desktop.shortcutKey.listenConflictResolved(() => {}); //=> listen to shortcut key conflict
resolved status

Desktop.shortcutKey.register([
  ...
], {
  ...
}, ...]); //=> registering shortcut keys actions

Desktop.shortcutKey.unregisterKeys('widget-one-example'); //=> used to unregister on unmount,
widgetElement used for register should be provided
```

- [Methods, on page 130](#)

Methods

register()

Registers the shortcut key actions.

Example

```
Desktop.shortcutKey.register([
  {
    componentName: "Sample Comp",
    actionName: "login",
    modifierKeys: "ctrlKey_altKey",
    key: "r",
    callback: (data: Service.shortcut.EKeyInfo) => {}
  }
])
```

The following table lists the payload details:

Name	Type	Description	Required
componentName	String	The name of the functionality, component, or the widget.	Yes
actionName	String	Name of the action or operation performed by the assigned shortcut keys.	Yes
modifierKeys	String	The modifier key is used commonly in keyboard shortcuts on the host platform. The keyboard modifier key combinations are: <ul style="list-style-type: none"> • Ctrl + Shift • Alt + Shift • Ctrl + Alt • Shift • Ctrl • Alt 	Yes
key	String	The main key to be combined with modifier keys. For example, Ctrl + Shift + e where Ctrl and Shift are the modifier keys, and e is the main key.	Yes
callback	Function	Client code can add the callback function while registering the shortcut key. When the shortcut key framework finds the matching keys from the registered keys list on the keyboard <code>keyup</code> event, then the shortcut framework invokes the callback method. The <code>keyup</code> event is invoked when a key is released.	Yes

Returns

{Array} The array of objects.

Example Response

```
const register = {
  [{
    "widgetElement": "agentx-wc-navigation",
    "group": "Navigation",
    "action": "Open Home Page",
    "modifierKeys": "ctrlKey_altKey",
    "key": "1"
  },
  // widgetElement is the name of the web component
  // group is the name of the functionality, component, or the widget.
  {
    "widgetElement": "agentx-wc-navigation",
    "group": "Navigation",
    "action": "Open Agent Performance Statistics Page",
    "modifierKeys": "ctrlKey_altKey",
    "key": "2"
  },
  {
    "widgetElement": "agentx-wc-navigation",
    "group": "Navigation",
    "action": "Open Widget Using JS API Page",
    "modifierKeys": "ctrlKey_altKey",
    "key": "3"
  }
  ]
}
```

getRegisteredKeys()

Retrieves the registered shortcut keys with the modifiers.

Example

```
console.log(Desktop.shortcutKey.getRegisteredKeys());
```

Returns

{Array} The array of objects.

Example Response

```
const getRegisteredKeys = [
  ["Go to Available StatectrlKey_altKey", {
    "widgetElement": "agentx-react-state-selector",
    "group": "Agent State",
    "action": "Go to Available State",
    "modifierKeys": "ctrlKey_altKey",
    "key": "r"
  }],
  ["Go to Idle StatectrlKey_altKey", {
    "widgetElement": "agentx-react-state-selector",
    "group": "Agent State",
    "action": "Go to Idle State",
    "modifierKeys": "ctrlKey_altKey",
    "key": "n"
  }],
  ["Send EmailctrlKey_altKeys", {
```

```

        "widgetElement": "agentx-react-email-composer",
        "group": "Email Handling",
        "action": "Send Email",
        "modifierKeys": "ctrlKey_altKey",
        "key": "s"
    }],
    ["ReplyctrlKey_shiftKey6", {
        "widgetElement": "agentx-react-email-composer",
        "group": "Email Handling",
        "action": "Reply",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "6"
    }],
    ["Reply AllctrlKey_shiftKey5", {
        "widgetElement": "agentx-react-email-composer",
        "group": "Email Handling",
        "action": "Reply All",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "5"
    }],
    ["Open Outbound callctrlKey_altKeyo", {
        "widgetElement": "agentx-react-out-dial",
        "group": "Outbound",
        "action": "Open Outbound call",
        "modifierKeys": "ctrlKey_altKey",
        "key": "o"
    }],
    ["Open Notification CenterctrlKey_altKeyi", {
        "widgetElement": "wagentx-wc-menu-notification",
        "group": "Notification",
        "action": "Open Notification Center",
        "modifierKeys": "ctrlKey_altKey",
        "key": "i"
    }],
    ["Enable Silent NotificationsctrlKey_altKeyd", {
        "widgetElement": "wagentx-wc-menu-notification",
        "group": "Notification",
        "action": "Enable Silent Notifications",
        "modifierKeys": "ctrlKey_altKey",
        "key": "d"
    }],
    ["Accept Chat/Email/SocialctrlKey_altKeya", {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Application",
        "action": "Accept Chat/Email/Social",
        "modifierKeys": "ctrlKey_altKey",
        "key": "a"
    }],
    ["Switch between PopoversctrlKey_altKey p", {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Application",
        "action": "Switch between Popovers",
        "modifierKeys": "ctrlKey_altKey",
        "key": "p"
    }],
    ["Expand/Collapse the PopoverctrlKey_shiftKey9", {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Application",
        "action": "Expand/Collapse the Popover",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "9"
    }],
    ["Accept all (visible) PopoversctrlKey_shiftKey4", {
        "widgetElement": "agentx-react-interaction-control",

```

```

        "group": "Application",
        "action": "Accept all (visible) Popovers",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "4"
    }],
    [
        "Hold/Resume CallctrlKey_altKeyv", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Hold/Resume Call",
            "modifierKeys": "ctrlKey_altKey",
            "key": "v"
        }
    ],
    [
        "Pause/Resume RecordingctrlKey_shiftKeyz", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Pause/Resume Recording",
            "modifierKeys": "ctrlKey_shiftKey",
            "key": "z"
        }
    ],
    [
        "Conference Request for Call/ChatctrlKey_altKeyh", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Conference Request for Call/Chat",
            "modifierKeys": "ctrlKey_altKey",
            "key": "h"
        }
    ],
    [
        "Consult Request for CallctrlKey_altKeyc", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Consult Request for Call",
            "modifierKeys": "ctrlKey_altKey",
            "key": "c"
        }
    ],
    [
        "End for All ChannelsctrlKey_altKeye", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "End for All Channels",
            "modifierKeys": "ctrlKey_altKey",
            "key": "e"
        }
    ],
    [
        "Transfer Request for All ChannelsctrlKey_altKeyx", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Transfer Request for All Channels",
            "modifierKeys": "ctrlKey_altKey",
            "key": "x"
        }
    ],
    [
        "Save Edited CAD Variable ValuesctrlKey_altKeym", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Save Edited CAD Variable Values",
            "modifierKeys": "ctrlKey_altKey",
            "key": "m"
        }
    ],
    [
        "Revert Edited CAD Variable ValuesctrlKey_altKeyz", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",
            "action": "Revert Edited CAD Variable Values",
            "modifierKeys": "ctrlKey_altKey",
            "key": "z"
        }
    ],
    [
        "Expand/CollapsectrlKey_shiftKey", {
            "widgetElement": "agentx-react-interaction-control",
            "group": "Interaction Control",

```

```

        "action": "Expand/Collapse",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "y"
    }],
    ["Wrap Up ReasonctrlKey_altKeyw", {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Wrap Up Reason",
        "modifierKeys": "ctrlKey_altKey",
        "key": "w"
    }],
    ["Open User ProfilectrlKey_altKeyu", {
        "widgetElement": "agentx-react-agent-profile",
        "group": "User Profile",
        "action": "Open User Profile",
        "modifierKeys": "ctrlKey_altKey",
        "key": "u"
    }],
    ["Sign OutctrlKey_altKeyl", {
        "widgetElement": "agentx-react-agent-profile",
        "group": "User Profile",
        "action": "Sign Out",
        "modifierKeys": "ctrlKey_altKey",
        "key": "l"
    }],
    ["Open Keyboard Shortcuts ListctrlKey_altKeyf", {
        "widgetElement": "agentx-react-agent-profile",
        "group": "User Profile",
        "action": "Open Keyboard Shortcuts List",
        "modifierKeys": "ctrlKey_altKey",
        "key": "f"
    }],
    ["Download Error ReportctrlKey_shiftKey2", {
        "widgetElement": "agentx-react-agent-profile",
        "group": "User Profile",
        "action": "Download Error Report",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "2"
    }],
    ["Switch between active tasksctrlKey_shiftKey8", {
        "widgetElement": "agentx-wc-task-list",
        "group": "Task List",
        "action": "Switch between active tasks",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "8"
    }],
    ["Expand/Collapse the Task PanelctrlKey_shiftKey7", {
        "widgetElement": "agentx-wc-task-list",
        "group": "Task List",
        "action": "Expand/Collapse the Task Panel",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "7"
    }],
    ["Open Navigation TabctrlKey_altKeyt", {
        "widgetElement": "agentx-wc-connector",
        "group": "Connector View",
        "action": "Open Navigation Tab",
        "modifierKeys": "ctrlKey_altKey",
        "key": "t"
    }],
    ["RefreshctrlKey_altKeyb", {
        "widgetElement": "agentx-wc-connector",
        "group": "Connector View",
        "action": "Refresh",

```

```

        "modifierKeys": "ctrlKey_altKey",
        "key": "b"
    }},
    ["Open HomectrlKey_altKey1", {
        "widgetElement": "agentx-wc-navigation",
        "group": "Navigation",
        "action": "Open Home",
        "modifierKeys": "ctrlKey_altKey",
        "key": "1"
    }},
    ["Open Agent Performance StatisticsctrlKey_altKey2", {
        "widgetElement": "agentx-wc-navigation",
        "group": "Navigation",
        "action": "Open Agent Performance Statistics",
        "modifierKeys": "ctrlKey_altKey",
        "key": "2"
    }},
    ["Open Webex Experience Manager MetricsctrlKey_altKey3", {
        "widgetElement": "agentx-wc-navigation",
        "group": "Navigation",
        "action": "Open Webex Experience Manager Metrics",
        "modifierKeys": "ctrlKey_altKey",
        "key": "3"
    }]
]

```

Default Shortcut Keys

Logs the default shortcut keys.

Example

```
console.log(Desktop.shortcutKey.DEFAULT_SHORTCUT_KEYS);
```

Returns

{Array} The array of objects.

Example Response

```

const DEFAULT_SHORTCUT_KEYS = {
  "agentState": [{
    "widgetElement": "agentx-react-state-selector",
    "group": "Agent State",
    "action": "Go to Available State",
    "modifierKeys": "ctrlKey_altKey",
    "key": "r"
  }, {
    "widgetElement": "agentx-react-state-selector",
    "group": "Agent State",
    "action": "Go to Idle State",
    "modifierKeys": "ctrlKey_altKey",
    "key": "n"
  }],
  "emailComposer": [{
    "widgetElement": "agentx-react-email-composer",
    "group": "Email Handling",
    "action": "Send Email",
    "modifierKeys": "ctrlKey_altKey",
    "key": "s"
  }, {
    "widgetElement": "agentx-react-email-composer",
    "group": "Email Handling",

```

```

        "action": "Reply",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "6"
    }, {
        "widgetElement": "agentx-react-email-composer",
        "group": "Email Handling",
        "action": "Reply All",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "5"
    }
  ],
  "outDial": [{
    "widgetElement": "agentx-react-out-dial",
    "group": "Outbound",
    "action": "Open Outbound call",
    "modifierKeys": "ctrlKey_altKey",
    "key": "o"
  }],
  "notification": [{
    "widgetElement": "wagentx-wc-menu-notification",
    "group": "Notification",
    "action": "Open Notification Center",
    "modifierKeys": "ctrlKey_altKey",
    "key": "i"
  }],
  {
    "widgetElement": "wagentx-wc-menu-notification",
    "group": "Notification",
    "action": "Enable Silent Notifications",
    "modifierKeys": "ctrlKey_altKey",
    "key": "d"
  }
],
  "interactionPopover": [{
    "widgetElement": "agentx-react-interaction-control",
    "group": "Application",
    "action": "Accept Chat/Email/Social",
    "modifierKeys": "ctrlKey_altKey",
    "key": "a"
  }],
  {
    "widgetElement": "agentx-react-interaction-control",
    "group": "Application",
    "action": "Switch between Popovers",
    "modifierKeys": "ctrlKey_altKey",
    "key": "p"
  },
  {
    "widgetElement": "agentx-react-interaction-control",
    "group": "Application",
    "action": "Expand/Collapse the Popover",
    "modifierKeys": "ctrlKey_shiftKey",
    "key": "9"
  },
  {
    "widgetElement": "agentx-react-interaction-control",
    "group": "Application",
    "action": "Accept all (visible) Popovers",
    "modifierKeys": "ctrlKey_shiftKey",
    "key": "4"
  }
],
  "interactionControl": [{
    "widgetElement": "agentx-react-interaction-control",
    "group": "Interaction Control",
    "action": "Hold/Resume Call",
    "modifierKeys": "ctrlKey_altKey",
    "key": "v"
  }],
  {
    "widgetElement": "agentx-react-interaction-control",
    "group": "Interaction Control",

```



```

        "action": "Pause/Resume Recording",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "z"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Conference Request for Call/Chat",
        "modifierKeys": "ctrlKey_altKey",
        "key": "h"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Consult Request for Call",
        "modifierKeys": "ctrlKey_altKey",
        "key": "c"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "End for All Channels",
        "modifierKeys": "ctrlKey_altKey",
        "key": "e"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Transfer Request for All Channels",
        "modifierKeys": "ctrlKey_altKey",
        "key": "x"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Save Edited CAD Variable Values",
        "modifierKeys": "ctrlKey_altKey",
        "key": "m"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Revert Edited CAD Variable Values",
        "modifierKeys": "ctrlKey_altKey",
        "key": "z"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Expand/Collapse",
        "modifierKeys": "ctrlKey_shiftKey",
        "key": "y"
    }, {
        "widgetElement": "agentx-react-interaction-control",
        "group": "Interaction Control",
        "action": "Wrap Up Reason",
        "modifierKeys": "ctrlKey_altKey",
        "key": "w"
    }
  ],
  "userProfile": [{
    "widgetElement": "agentx-react-agent-profile",
    "group": "User Profile",
    "action": "Open User Profile",
    "modifierKeys": "ctrlKey_altKey",
    "key": "u"
  }, {
    "widgetElement": "agentx-react-agent-profile",
    "group": "User Profile",
    "action": "Sign Out",
    "modifierKeys": "ctrlKey_altKey",
    "key": "l"
  }
]

```

```

    }, {
      "widgetElement": "agentx-react-agent-profile",
      "group": "User Profile",
      "action": "Open Keyboard Shortcuts List",
      "modifierKeys": "ctrlKey_altKey",
      "key": "f"
    }, {
      "widgetElement": "agentx-react-agent-profile",
      "group": "User Profile",
      "action": "Download Error Report",
      "modifierKeys": "ctrlKey_shiftKey",
      "key": "2"
    }
  ],
  "taskList": [{
    "widgetElement": "agentx-wc-task-list",
    "group": "Task List",
    "action": "Switch between active tasks",
    "modifierKeys": "ctrlKey_shiftKey",
    "key": "8"
  }, {
    "widgetElement": "agentx-wc-task-list",
    "group": "Task List",
    "action": "Expand/Collapse the Task Panel",
    "modifierKeys": "ctrlKey_shiftKey",
    "key": "7"
  }
  ],
  "connectorView": [{
    "widgetElement": "agentx-wc-connector",
    "group": "Connector View",
    "action": "Open Navigation Tab",
    "modifierKeys": "ctrlKey_altKey",
    "key": "t"
  }, {
    "widgetElement": "agentx-wc-connector",
    "group": "Connector View",
    "action": "Refresh",
    "modifierKeys": "ctrlKey_altKey",
    "key": "b"
  }
  ]
}

```

Modifiers

Logs the shortcut key modifier. A modifier shortcut key is Shift, Ctrl, Alt, or a combination of these keys.

Example

```
console.log(Desktop.shortcutKey.MODIFIERS);
```

Returns

{Object} The object response.

Example Response

```

const MODIFIERS = {
  "CTRL_SHIFT": "ctrlKey_shiftKey",
  "ALT_SHIFT": "altKey_shiftKey",
  "CTRL_ALT": "ctrlKey_altKey",
  "SHIFT": "shiftKey",
  "CTRL": "ctrlKey",
  "ALT": "altKey"
}

```

Registered Keys

Logs registered shortcut keys without the modifiers.

Example

```
console.log(Desktop.shortcutKey.REGISTERED_KEYS);
```

Returns

{Object} The object response.

Example Response

```
const REGISTERED_KEYS = {
  "EXPAND_COLLAPSE_INTERACTION_PANEL_KEY": "y",
  "SAVE_EDITED_CAD_KEY": "m",
  "REVERT_EDITED_CAD_KEY": "z",
  "HOLD_RESUME_CALL_KEY": "v",
  "TRANSFER_KEY": "x",
  "CONSULT_KEY": "c",
  "END_KEY": "e",
  "CONFERENCE_KEY": "h",
  "PAUSE_RESUME_RECORDING_KEY": "z",
  "GO_TO_AVAILABLE_KEY": "r",
  "OPEN_STATE_SELECTOR_KEY": "n",
  "SEND_EMAIL_KEY": "s",
  "REPLY_EMAIL_KEY": "6",
  "REPLY_ALL_EMAIL_KEY": "5",
  "OPEN_USER_PROFILE_KEY": "u",
  "ENABLE_SILENT_NOTIFICATION_KEY": "d",
  "OPEN_SHORTCUT_KEY_MODAL_KEY": "f",
  "DOWNLOAD_ERROR_REPORT_KEY": "2",
  "SIGNOUT_KEY": "1",
  "ACCEPT_TASK_KEY": "a",
  "SWITCH_POPOVER_KEY": "p",
  "EXPAND_COLLAPSE_POPOVER_KEY": "9",
  "OPEN_OUTDIAL_KEY": "o",
  "OPEN_WRAP_UP_KEY": "w",
  "EXPAND_COLLAPSE_TASK_LIST_PANEL_KEY": "7",
  "OPEN_NOTIFICATION_CENTER_KEY": "i",
  "OPEN_NAVIGATION_TAB_KEY": "t",
  "REFRESH_KEY": "b",
  "SWITCH_TASK_KEY": "8",
  "ACCEPT_ALL_TASK_KEY": "4"
}
```

unregisterKeys()

Unregisters the shortcut keys.

Example

```
Desktop.shortcutKey.unregisterKeys([
  {
    widgetElement: "custom-element",
    group: "custom widget",
    action: "login",
    modifierKeys: "ctrlKey_altKey",
    key: "r"
  },
  {
    widgetElement: "custom-element",
```

```

        group: "custom widget",
        action: "logout",
        modifierKeys: "ctrlKey_altKey",
        key: "n"
      }
    ]
  })
}

```

listenKeyPress(event)

Listens to shortcut key press.

Example

```

Desktop.shortcutKey.listenKeyPress(event => {
  console.log("JSSDK ShortcutKey listenKeyPress: ", event);
});

```

Parameters

Name	Type	Description	Required
event	Object	The event occurs when a keyboard key is released.	Yes

Returns

{Object} The object response.

Example Response

```

{
  "type": "execute",
  "data": {
    "widgetElement": "agentx-react-agent-profile",
    "group": "User Profile",
    "action": "Open Keyboard Shortcuts List",
    "modifierKeys": "ctrlKey_altKey",
    "key": "f",
    "keyCombination": "Ctrl + Alt + F"
  },
  "keyboardEvent": {
    "isTrusted": true
  }
}

```

listenKeyConflict(event)

Listens to shortcut key conflicts.

Example

```

Desktop.shortcutKey.listenKeyConflict((event => {
  console.log("JSSDK ShortcutKey listenKeyPress: ", event);
}));

```

Parameters

Name	Type	Description	Required
event	Object	The event occurs when a keyboard key is released.	Yes

Returns

{Object} The object response.

Example Response

```
{
  "type": "conflict",
  "data": {
    "widgetElement": "agentx-react-agent-profile",
    "group": "User Profile",
    "action": "Open Keyboard Shortcuts List",
    "modifierKeys": "ctrlKey_altKey",
    "key": "f",
    "isConflict": true
  }
}
```

listenConflictResolved()

Listens to shortcut key conflict resolved status.

Example

```
Desktop.shortcutKey.listenConflictResolved(() => {});
```

`listenConflictResolved()`



PART III

Migration

- [Finesse Gadget Migration](#), on page 145



CHAPTER 13

Finesse Gadget Migration

- Migrate Finesse Embedded Web Application Gadgets, on page 145
- Migrate JavaScript Based Finesse Gadgets, on page 147

Migrate Finesse Embedded Web Application Gadgets

You can migrate a Finesse embedded web application gadget into a Webex Contact Center Agent Desktop iFrame based widget.



Note

Conversion of a Finesse gadget depends on the web application configuration—whether it could be loaded in an iFrame or not.

Finesse—Embedded Web Application Sample Gadget

The embedded web application sample gadget displays a web page in an iFrame within the gadget. It is intended to serve as an example of placing an external web application in the Finesse gadget.

For more information on the embedded web application sample gadget, see <https://github.com/CiscoDevNet/finesse-sample-code/tree/master/EmbeddedWebAppSampleGadget>.

Webex Contact Center Agent Desktop—agentx-wc-iframe

Desktop has the inbuilt functionality (Web Component) to load web applications in an iFrame. Using the component `agentx-wc-iframe` within the desktop layout, you can load web applications in the horizontal header, navigation bar, and auxiliary information pane.

For more information on Webex Contact Center Agent Desktop widgets, see <https://github.com/CiscoDevNet/webex-contact-center-widget-starter/tree/master/Examples>.

The following are the Finesse embedded gadget files:

- `EmbeddedWebApp.css`
- `EmbeddedWebApp.js`
- `EmbeddedWebApp.xml`

The `EmbeddedWebApp.xml` is the main file that contains the URL of the web application.

Before you begin

- Understand the `script type` details from the Finesse embedded gadget file `EmbeddedWebApp.xml`.

```
<script type="text/javascript">
    // initialize the gadget running the init handler defined in
    EmbeddedWebApp.js
    gadgets.HubSettings.onConnect = function () {
        finesse.modules.EmbeddedWebAppGadget.init("https://www.bing.com");
    };
</script>
```

- Understand the JSON layout format. For more information on JSON layout, see the *Desktop Layout* section in the *Provisioning* chapter of the [Cisco Webex Contact Center Setup and Administration Guide](#).

Step 1 Access the JSON layout file from the Webex Contact Center Management Portal.

Step 2 In the **comp** property tag, enter the component value as **agentx-wc-iframe**.

```
{
  "comp": "agentx-wc-iframe"
},
```

Step 3 In the **src attributes** property tag, enter the URL of the web application.

```
{
  "comp": "agentx-wc-iframe",
  "attributes": {
    "src": "https://www.bing.com"
  },
```

Step 4 In the **wrapper** property tag, enter the title and the maximize area name.

```
{
  "comp": "agentx-wc-iframe",
  "attributes": {
    "src": "https://www.bing.com"
  },
  "wrapper": {
    "title": "AgentX iFrame",
    "maximizeAreaName": "app-maximize-area"
  },
```

Step 5 Use the **style** property tag to resize the iFrame.

Note The iFrame within the Desktop supports native iFrame properties.

```
{
  "comp": "agentx-wc-iframe",
  "attributes": {
    "src": "https://www.bing.com"
  },
  "wrapper": {
    "title": "AgentX iFrame",
    "maximizeAreaName": "app-maximize-area"
  },
  "style": {
    "height": 504 px,
    "width": 520 px,
    "display": "inline-block",
    "align-items": "center"
  },
```

Step 6 Use the **attributes** property tag to add iFrame attributes.

```
{
  "comp": "agentx-wc-iframe",
  "attributes": {
    "src": "https://www.bing.com"
    "sandbox": "allow-scripts allow popups"
  },
  "wrapper": {
    "title": "AgentX iFrame",
    "maximizeAreaName": "app-maximize-area"
  },
  "style": {
    "height": 504 px,
    "width": 520 px,
    "display": "inline-block",
    "align-items": "center"
  },
}
```

Migrate JavaScript Based Finesse Gadgets

There is no straightforward migration from a Shindig based gadget to a wxcc-desktop based widget. As the Shindig gadget and the wxcc-desktop widget leverage technologies, you can refactor the existing source code to develop wxcc-desktop based widgets. Finesse gadgets are based on the Apache Shindig gadget specification. For more information on Apache Shindig, see <https://shindig.apache.org/>. You can use any JavaScript library, framework, or our vanilla JavaScript to build a UI based gadget. The final content must be embedded within the gadget XML or HTML.

Finesse Gadgets

Static Resources—Finesse gadgets consist of building blocks such as XML, HTML, CSS, JavaScript, and images. The gadgets are deployed in the Finesse 3rdpartygadget web application.

Elements—The following are the important elements of a gadget:

- The `<Module>` tag indicates that the XML file contains a gadget which is a root level element.
- The `<ModulePrefs>` tag contains information about the gadget. The tag includes the title, description, author, other optional features, and configuration related to the gadget from the Finesse desktop container.
- The `<Content type="html">` header tag indicates that the gadget's content type is HTML. The tag includes all markup content (HTML) and static resources that are embedded within the content element.

Example: Sample Gadget

```
<Module>
  <ModulePrefs title="X-Counter" description="X-Counter Example"></ModulePrefs>
  <Content type="html">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"/>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment.min.js"></script>
    <![CDATA[
      <style>
        button, p {
          display: inline-block;
```

```

    }
    </style><button aria-label="decrement"></button><p>0</p><button
aria-label="increment"></button><script>
    var valueElement = document.querySelector('p');
    function increment() {
        const value = valueElement.innerText;
        valueElement.innerText = parseInt(value) + 1;
    }
    function decrement() {
        const value = valueElement.innerText;
        valueElement.innerText = parseInt(value) - 1;
    }
    var incrementButton = document.querySelectorAll('button')[1];
    var decrementButton = document.querySelectorAll('button')[0];

    incrementButton.addEventListener('click', e => increment);

    decrementButton.addEventListener('click', e => decrement);
</script>
]]>
</Content>
</Module>

```

For more information on Cisco Finesse gadgets, see <https://developer.cisco.com/docs/finesse/#!/finesse-overview/cisco-finesse-gadgets-finesse-javascript-library-api>.

Webex Contact Center Agent Desktop Widgets

Desktop widget is based on the micro-frontend approach that leverages the browser's native Web Component module.

Static Resources—Custom widgets must bundle all their static resources (CSS, JavaScript, markup, and images) in a single JavaScript file using a bundle library (for example, Webpack, Rollup, Parcel). The bundle must be hosted in a Content Delivery Network (CDN). Custom widgets can use any JavaScript framework to develop Web Components.

Shared Data from Desktop (for example, ModulePrefs)—Custom widgets can have shared data or configuration from Desktop as a Web Component property or attributes.



Note

Desktop shared data or configuration is different from Finesse gadget shared data or configuration. The developer must understand the relevant shared data or configuration that are available in `wxcc-desktop`. For more information, see [Data Provider—Widget Properties and Attributes](#), on page 8.

The JavaScript bundle must have the root element as Web Component, and the element name must be mentioned within the desktop layout configuration. Custom widgets can be developed using any web-based library or framework, for example, React, Angular, or Web Component. Custom widgets can also load additional static resources such as CSS, JavaScript, Markup, and images internally using Ajax or markup tags (script, link, img, and so on).

The following is an example of the sample Hello World widget without any bundling library (helloworld.js) and build using vanilla JavaScript. The widget name is `<x-counter>`.

Example: Sample Widget Without Any Bundling Library

```

const template = document.createElement('template');
template.innerHTML = `
  <style>
    button, p {

```

```

        display: inline-block;
    }
</style>
<button aria-label="decrement">-</button>
    <p>0</p>
<button aria-label="increment">+</button>
`
;

class XCounter extends HTMLElement {

    increment() {
        const value = this.valueElement.innerText;
        this.valueElement.innerText = parseInt(value) + 1;
    }
    decrement() {
        const value = valueElement.innerText;
        valueElement.innerText = parseInt(value) - 1;
    }
    constructor() {
        super();

        this.attachShadow({
            mode: 'open'
        });
        this.shadowRoot.appendChild(template.content.cloneNode(true));

        this.valueElement = this.shadowRoot.querySelector('p');
        this.incrementButton = this.shadowRoot.querySelectorAll('button')[1];
        this.decrementButton = this.shadowRoot.querySelectorAll('button')[0];

        this.incrementButton.addEventListener('click', e => this.increment);

        this.decrementButton.addEventListener('click', e => this.decrement);
    }
}

customElements.define('x-counter', XCounter);

```

