



## 5.3 Simplified Desktop .NET API Specification

### Bright Pattern Documentation

Generated: 7/03/2022 9:03 am

Content is available under license unless otherwise noted.

## Table of Contents

Table of Contents	2
Purpose	4
Audience	4
General Information	4
Event Delivery	4
API Instantiation and Initialization	5
API Methods	6
InitAPI	6
Syntax	6
ShutdownAPI	6
Syntax	6
CallDial	6
Syntax	6
MuteCallRecording	6
Syntax	6
UnmuteCallRecording	7
Syntax	7
MuteScreenRecordings	7
Syntax	7
UnmuteScreenRecordings	7
Syntax	7
CallSetScenarioVariable	7
Syntax	7
CallSingleStepTransfer	7
Syntax	8
API Events	8
evtApiUp	8
Syntax	8
evtApiDown	8
Syntax	8
evtCallDialing	8
Syntax	8
evtCallOffered	8
Syntax	9
evtCallDisconnected	9
Syntax	9
onError	9
Syntax	9
onCallRecordingStarted	9
Syntax	9
onCallRecordingStopped	9
Syntax	9
onCallRecordingMuted	10
Syntax	10
onCallRecordingUnmuted	10
Syntax	10
onScreenRecordingStarted	10
Syntax	10
onScreenRecordingCompleted	10
Syntax	10
onScreenRecordingsMuted	10
Syntax	11
onScreenRecordingsUnmuted	11
Syntax	11
API Properties	11
connected	11
Type	11
calls	11
Type	11

<b>Auxiliary Classes</b>	<b>11</b>
DesktopControlAPI.BaseEventArgs	12
Members	12
DesktopControlAPI.Call	12
Members	12
DesktopControlAPI.CallArgs	12
Members	12
DesktopControlAPI.ErrorEventArgs	13
Members	13
DesktopControlAPI.ScreenRecordingArgs	13
Members	13

# Purpose

The Bright Pattern Contact Center *Simplified Desktop .NET API Specification* describes the methods and events of the Simplified Desktop .NET API, which provides access to a number of functions of the Bright Pattern Contact Center Agent Desktop application from .NET-based third-party applications.

For more information about the Agent Desktop application, see the Bright Pattern Contact Center [Agent Guide](#).

# Audience

This guide is intended for the IT personnel responsible for the data infrastructure of Bright Pattern Contact Center-based contact centers. Readers of this guide are expected to have expertise in web application development as well as a solid understanding of contact center operations.

# General Information

The API supports the following functions:

- Making calls
- Getting notifications about initiated calls
- Getting notifications about incoming calls
- Getting notifications when both outgoing and incoming calls are released
- Pausing and resuming voice and screen recordings
- Getting notifications when recording is stopped and started
- Making single-step call transfers
- Setting scenario variables

The API itself does not replace the softphone functionality; the Bright Pattern Contact Center Agent Desktop application with a softphone (or hardphone) must be running on the user's computer in order for the API to function.

The API communicates with Agent Desktop locally, and therefore, an Agent Desktop session must be started and the softphone plugin (Agent Desktop Helper Application) must be active.

The API uses the identity of the user logged into the Agent Desktop application in the current Windows session.

# Event Delivery

Communications with the softphone plugin are handled by the lowest level of the API; all transport is handled by the .NET platform and all transport-related events are delivered on threads maintained by the platform. Note that direct updating of the UI (either Windows Forms, or WPF) is only permitted on the UI thread--in most cases, the main thread of the application. Although different mechanisms exist to convey execution flow from non-UI to UI threads, the API has its own mechanism that allows receiving events emitted by the API classes on the UI thread--event dispatcher classes.

Each API class has a child class named *eventDispatcher* (referred to as "event dispatcher"). The event dispatcher class intercepts and re-emits all events emitted by its parent class, while guaranteeing that the events will be delivered on the thread on which the API has been created. Event handlers registered with the API's event dispatcher class created on the UI thread may directly update elements of the application's UI.

All events emitted by all objects follow Microsoft's guideline that asks that all event handlers take two parameters: the object that has emitted the event and another object that encapsulates all arguments of the event. Such an argument object should inherit the *System.EventArgs* class and provide any additional data or capabilities in the form of additional properties and methods. All events emitted by the event dispatcher objects carry the reference to the original sender object.

## API Instantiation and Initialization

First, the client should create an *Api* object and attach event handlers:

```
m_api = new DesktopControlAPI.ApiLite();  
m_api.eventDispatcher.evtApiUp += this.onApiUp;  
m_api.eventDispatcher.evtApiDown += this.onApiDown;  
m_api.eventDispatcher.evtCallOffered += this.onCallOffered;  
m_api.eventDispatcher.evtCallDialing += this.onCallDialing;  
m_api.eventDispatcher.evtCallDisconnected += this.onCallDisconnected;
```

Note that by default API logging is enabled:

```
ApiLite(bool _enableLog = true);
```

To disable logging, set `enableLog` to `false`.

```
m_api = new DesktopControlAPI.ApiLite(false);
```

Once API is created and event handlers are attached, the client may initialize the API instance:

```
m_api.InitAPI();
```

Once API is connected to the softphone and the `evtApiUp` event is received, the client may dial calls:

```
m_api.CallDial("4154556565");
```

When application terminates the following method should be called to ensure clean termination of the TCP connection between API and the Agent Desktop:

```
m_api.ShutdownAPI();
```

## API Methods

### InitAPI

*InitAPI* initializes the API.

This request initializes the TCP connection procedure. Once the connection is established, the evtApiUp event is sent to the .NET application.

#### Syntax

```
bool InitAPI();
```

### ShutdownAPI

*ShutdownAPI* terminates the TCP connection between the API and the Agent Desktop.

#### Syntax

```
void ShutdownAPI();
```

### CallDial

*CallDial* initiates a call to the number specified in *destination*.

Specification of the [global interaction identifier](#) (GIID) is optional. This returns request ID.

#### Syntax

```
UInt64 CallDial(string destination, string giid);
```

### MuteCallRecording

*MuteCallRecording* mutes audio recording of the call identified by *callId* on the given desktop.

The recording will continue, but any voice signal will be replaced with silence.

#### Syntax

```
UInt64 MuteCallRecording(string callId);
```

## UnmuteCallRecording

*UnmuteCallRecording* resumes previously muted audio recording for the call identified by *callId* on the given desktop.

### Syntax

```
UInt64 UnmuteCallRecording(string callId);
```

## MuteScreenRecordings

*MuteScreenRecordings* mutes screen recording on the given desktop.

For the period when screen recording is muted, the recording will contain a static snapshot of the desktop at the moment when mute was applied.

This request is processed by the Agent Desktop Helper Application.

### Syntax

```
UInt64 MuteScreenRecordings();
```

## UnmuteScreenRecordings

*UnmuteScreenRecordings* unmutes previously muted screen recording on the given desktop.

This request is processed by the Agent Desktop Helper Application.

### Syntax

```
UInt64 UnmuteScreenRecordings();
```

## CallSetScenarioVariable

*CallSetScenarioVariable* sets a variable identified by *name* for the scenario associated with the call identified by *callId* to the value specified in *value*.

### Syntax

```
UInt64 CallSetScenarioVariable(string _callId, string _name, string _value);
```

## CallSingleStepTransfer

This method initiates a [single-step \(blind\) transfer](#) of the call identified by *callId* to the number specified in *dest*.

## Syntax

```
UInt64 CallSingleStepTransfer(string _callId, string _dest);
```

# API Events

## evtApiUp

*evtApiUp* is sent when the API is successfully connected (or reconnected) to the softphone.

Note that no information is provided about calls that already may be present on the Agent Desktop.

## Syntax

```
void evtApiUp(System.EventArgs args);
```

## evtApiDown

*evtApiDown* is sent when the ShutdownAPI method is called or when the API loses connection to the softphone.

In case of connection loss, the API will try to reconnect every 15 seconds; once reconnected, the *evtApiUp* event is sent.

## Syntax

```
void evtApiDown(System.EventArgs args);
```

## evtCallDialing

*evtCallDialing* is sent when the softphone starts dialing a call.

Note that this event reports calls both initiated by a prior [CallDial](#) method and dialed directly from the Agent Desktop.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

## Syntax

```
void evtCallDialing(DesktopControlAPI.CallArgs args);
```

## evtCallOffered



*evtCallOffered* is sent when the softphone receives a new incoming call.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

### Syntax

```
void evtCallOffered(DesktopControlAPI.CallArgs args);
```

## evtCallDisconnected

*evtCallDisconnected* is sent when an existing call is released.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

### Syntax

```
void evtCallDisconnected(DesktopControlAPI.CallArgs args);
```

## onError

*onError* is sent when an error occurs while executing a method.

For parameters, see the description of auxiliary class [DesktopControlAPI.ErrorEventArgs](#).

### Syntax

```
void onError(DesktopControlAPI. ErrorEventArgs args);
```

## onCallRecordingStarted

*onCallRecordingStarted* is sent when call recording is started. Note that this event does not indicate that a call was established or if a call was not recorded.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

### Syntax

```
void onCallRecordingStarted(DesktopControlAPI.CallArgs args);
```

## onCallRecordingStopped

*onCallRecordingStopped* is sent when call recording is stopped.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

### Syntax

```
void onCallRecordingStopped(DesktopControlAPI.CallArgs args);
```

## **onCallRecordingMuted**

*onCallRecordingMuted* is sent when call recording is muted.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

### **Syntax**

```
void onCallRecordingMuted(DesktopControlAPI.CallArgs args);
```

## **onCallRecordingUnmuted**

*onCallRecordingUnmuted* is sent when call recording is unmuted.

For parameters, see the description of auxiliary class [DesktopControlAPI.CallArgs](#).

### **Syntax**

```
void onCallRecordingUnmuted(DesktopControlAPI.CallArgs args);
```

## **onScreenRecordingStarted**

*onScreenRecordingStarted* is sent when screen recording is started.

For parameters, see the description of auxiliary class [DesktopControlAPI.ScreenRecordingArgs](#).

### **Syntax**

```
void onScreenRecordingStarted(DesktopControlAPI.ScreenRecordingArgs args);
```

## **onScreenRecordingCompleted**

*onScreenRecordingCompleted* is sent when screen recording is stopped.

For parameters, see the description of auxiliary class [DesktopControlAPI.ScreenRecordingArgs](#).

### **Syntax**

```
void onScreenRecordingCompleted(DesktopControlAPI.ScreenRecordingArgs args);
```

## **onScreenRecordingsMuted**

*onScreenRecordingsMuted* is sent when screen recording is muted.

For parameters, see the description of auxiliary class [DesktopControlAPI.BaseEventArgs](#).

## Syntax

```
void onScreenRecordingsMuted(DesktopControlAPI.BaseEventArgs args);
```

## onScreenRecordingsUnmuted

*onScreenRecordingsUnmuted* is sent when screen recording is unmuted.

For parameters, see the description of auxiliary class [DesktopControlAPI.BaseEventArgs](#).

## Syntax

```
void onScreenRecordingsUnmuted (DesktopControlAPI.BaseEventArgs args);
```

# API Properties

## connected

This is used to check if the API is connected to Agent Desktop.

It is True if evtApiUp was called before. It is False if evtApiDown was called or evtApiUp was never called.

## Type

boolean

## calls

This is used to collect calls on the agent's phone (there may be more than one call).

See the description of [auxiliary class](#) DesktopControlAPI.Call.

## Type

System.Collections.Generic.Dictionary

<string,DesktopControlAPI.Call>

# Auxiliary Classes

## DesktopControlAPI.BaseEventArgs

Base event arguments class

### Members

- **reqId** – Request ID returned by a call that caused the error

## DesktopControlAPI.Call

Describes a single phone call

### Members

- **m\_id** – Call ID; string
- **m\_direction** – Call direction; enum; (DesktopControlAPI.CallDirection.Inbound for incoming calls; DesktopControlAPI.CallDirection.Outbound for outgoing calls)
- **m\_ANI** – Caller ID number for incoming calls or agent's phone number for outgoing calls; string
- **m\_DNIS** – Number dialed by the calling party for incoming calls or number dialed by the agent for outgoing calls; string
- **m\_callerName** – Caller ID text name for incoming calls or agent's display name for outgoing calls; string
- **m\_serviceName** – Name of the service associated with this call; string
- **m\_agentId** – Agent's username; string
- **m\_agentPhone** – Agent's phone number; string

## DesktopControlAPI.CallArgs

Class for call attributes used in all call-related events

### Members

- **callId** – Call ID; string
- **requestId** – Identifier of the request related to this event; string
- **ANI** – Caller ID number for incoming calls or agent's phone number for outgoing calls; string
- **DNIS** – Number dialed by the calling party for incoming calls or number dialed by the agent for outgoing calls
- **callerName** – Caller ID text name for incoming calls or agent's display name for outgoing calls
- **serviceName** – Name of the service associated with this call
- **attachedData** – Key-value list of the data attached to this call by the scenario that processed it; scenario should use the [Attached Data](#) block for this purpose

- **requestId** - Optional request ID of the method to which the event responds
- **globalInteractionId** - [Global interaction identifier](#) (GIID); string

## DesktopControlAPI.ErrorEventArgs

Error event arguments class

### Members

- **reqId** – Request ID of the method that caused the error
- **callId** – Call ID; string
- **code** - Error code
- **description** - Error description

## DesktopControlAPI.ScreenRecordingArgs

Arguments of events related to screen recording

### Members

- **sessionId** – Screen recording session ID (starting from Bright Pattern Contact Center version 3.9, all agent activities, including interaction handling, are recorded via a single screen recording session; that is, this parameter has no purpose other than backward compatibility)
- **requestId** - Optional request ID of the method to which the event responds