

# Overview

The AgentWorkspace JS API interacts directly with Euphoria browser phone and agent workspace and allows a user to use Euphoria Agent workspace outside of Euphoria platform

# Usage Requirements

* Library Key is required to use this API. Please contact Euphoria if one is needed.
* JQuery Library is required for the library (version jquery-2.1.4.is recommended)
* The live AgentWorkspace Bundle JS must be included in your HTML page. Link:  
  <https://s3-eu-west-1.amazonaws.com/static.euphoria.co.za/Libraries/AgentWorkspaceBundle/AgentWorkspaceBundle.min.js>
* This Library relies on WebRTC as the main and only engine, hence it requires running in a browser that supports this functionality.

# Phone API

The Phone API is used to take control of the phone registration and all calls related actions.

## Commands

### Start

By using this command the following actions will be executed sequentially:

* Phone loading
* Agent login
* Phone registration

let authentication = {

Tenant: "<Your\_Tenant\_Name>",

ApiKey: "<Your\_API\_Key>",

LibraryKey: "<Your\_Library\_Key>"

}

let euphoriaPhoneObj = {

WssServer: "<PBX\_Server\_URL>", //pbx999.euphoria.co.za

SipUsername: "<Sip\_Username>",

SipPassword: "<Sip\_Password>",

$StatusBox: $("#stBox")

}

Phone.Start(euphoriaPhoneObj, authentication);

| Parameter | Description | Attributes |
| --- | --- | --- |
| authentication | Contains your API and library credentials | * Tenant: Your tenant name * ApiKey: Your API Key * LibraryKey: Library key to allow you use the library |
| euphoriaPhoneObj | Contains phone registration details | * WssServer: Your PBX server * SipUsername: SIP username of the extension to register * SipPassword: SIP password of the extension to register * $StatusBox: JQuery selector of the status box that will be used to show the phone statuses |

### Dial

Allows a manual outbound call to be made.

Phone.Dial("08000000000")

### AnswerCall

Accepts an incoming call

Phone.AnswerCall()

### HangUpCall

Terminates the current call

Phone.HangUpCall()

### CancelCall

Allows an early hangup of the current call

Phone.CancelCall()

### HoldCall

Allows the current call to be placed hold

Phone.HoldCall()

### UnHoldCall

Removes the hold status from the current call

Phone.UnHoldCall()

### MuteCall

Mute the microphone

Phone.MuteCall()

### UnMuteCall

Unmute the microphone

Phone.UnMuteCall()

### ChangeCallVolume

Change the volume of the current call

Phone.ChangeCallVolume(audioLevel)

audioLevel: integer value 0-100

### BlindTransfer

Blind transfer a call to a number

Phone.BlindTransfer(targetNumber)

targetNumber: phone number to transfer the current call to

### AttendedTransfer

Make a second call in order to transfer the current active line to. It returns an object

containing two functions (Confirm and Cancel) which allows the user to confirm or cancel

the transfer.

var transfer = Phone.AttendedTransfer(targetNumber)  
// returns {Confirm<function()>, Cancel()<function>}

targetNumber: phone number to transfer the current call to

### DtmfSend

Send a DTMF character (char) to the current call line

Phone.DtmfSend(characterToSend)

characterToSend: number 0-9, \*, # are allowed

### CalLine Object

Each call has a call line object that contains all the details of that call as attributes.

| Attribute/Function | Description |
| --- | --- |
| secsDuration (Integer) | Duration in seconds |
| crmTag | Can be used to set the CRM tag for the call.  callLine.crmTag = “CRM TAG”; |
| from | Source of the call |
| to | Destination of the call |
| direction | Direction of the call (incoming/outgoing) |
| sipCallID | SIP unique Id of the call |
| uniqueId | System generated unique ID of the call |
| isDialer (Bool) | Dialer flag indicate whether the call is dialer call or not |
| dispoEnabled (Bool) | Indicate whether the call requires disposition or not |
| submitDisposition(newDisposition, newCallOutcome, callbackDate) | Submit a disposition for the line. This function is very important for dialer calls: it allows the agents to get the next call when the current call disposition gets submitted as the system expects a disposition after those calls.  newDisposition: Success,Unsure, Failed, Failed Contact. newCallOutcome: The outcome of the call. callbackDate (optional): callback date if the call has been marked as callback. Format: “yyyy-MM-dd HH:mm:ss”. |
| contact | Contains all the dialer lead related details. |
| powerDialer (bool) | Identifies whether the call is a power dialer call |
| autoDialer (bool) | Identifies whether the call is an auto dialer call |
| localStartTime(string) | Start time of the call |
| completed (bool) | Identifies if a call is completed and disposition has been submitted if necessary |

### GetCurrentLine

Returns the call line of the current call

Phone.GetCurrentLine()

### GetCallLines

Returns all call lines available since registration

Phone.GetCallLines()

## Events (Phone.Events)

* **OnLoad**

Triggers once the phone is ready to use

Phone.Events.OnLoad(callbackFunction);

* **OnRegister**

Triggers on phone registration

Phone.Events.OnRegister(callbackFunction);

* **OnUnregister**

Triggers on phone deregistration

Phone.Events.OnUnregister(callbackFunction);

* **OnRinging**

Triggers on inbound ringing

Phone.Events.OnRinging(callbackFunction);

* **OnDialError**

Triggers if there was a dialing error

Phone.Events.OnDialError(callbackFunction);

# 

* **OnCallAccept**

Triggers on call answer (In/Out)

Phone.Events.OnCallAccept(callbackFunction);

* **OnDial**

Triggers before dialing the number

Phone.Events.OnDial(callbackFunction);

* **OnAfterDial**

Triggers after dialing the number

Phone.Events.OnAfterDial(callbackFunction);

* **OnCallTermination**

Triggers once the call is terminated

Phone.Events.OnLoad(callbackFunction);

* **OnIncomingCall**

Triggers on incoming call

Phone.Events.OnIncomingCall(callbackFunction);

* **OnCallFailure**

Triggers if dialing failed

Phone.Events.OnCallFailure(callbackFunction);

* **OnHold**

Triggers once the call is placed on hold

Phone.Events.OnHold(callbackFunction);

* **OnUnHold**

Triggers once the call is active after placed on hold

Phone.Events.OnUnHold(callbackFunction);

* **OnMute**

Triggers on microphone mute

Phone.Events.OnMute(callbackFunction);

* **OnUnMute**

Triggers on microphone unmute

Phone.Events.OnUnMute(callbackFunction);

* **OnVolumeChange**

Triggers on call volume change

Phone.Events.OnVolumeChange(callbackFunction);

* **OnCallTransfer**

Triggers after call transfer

Phone.Events.OnCallTransfer(callbackFunction);

# AgentWorkspace API

The AgentWorkspace API can be used to take control of agent related actions like pause, unpause & logout.

## Commands

### Pause

Allows the user to pause the agent for the reason selected

AgentWorkspace.Pause(pauseReason);

pauseReason: (string) The reason why you pausing the agent

### UnPause

Allows the user to un-pause the agent

AgentWorkspace.UnPause();

### LogOff

Allows the user to log the agent off the agent workspace

AgentWorkspace.LogOff();

### SkipPreptime

Allows the user to skip the preparation time for the preview dialer calls

AgentWorkspace.SkipPreptime();

## Events (AgentWorkspace.Events)

* **OnDialerLeadAvailable**

Triggers when a dialer lead is available and returns all the information related to the lead

AgentWorkspace.Events.OnDialerLeadAvailable(function(lead){

*// lead = {*

*// id: leadId,*

*// number: num, // PD*

*// numbers: contactNumbers, // PR*

*// crmTag: crmTag,*

*// campaignId: campaignId,*

*// campaignName: campaignName,*

*// campaignType: campaignType,*

*// listName: listName*

*// notes: notes,*

*// address: address,*

*// language: language,*

*// title: title,*

*// name: fullName,*

*// queueName: queueName*

*// queueType: queueType*

*// email: email*

*// }*

);

# Full Example

<**html**>

<**head**>

<**script** type="text/javascript" src="https://dt46w9nqlye04.cloudfront.net/Libraries/jQuery-2.1.4/jquery-2.1.4.min.js"></**script**>

<**script** type="text/javascript" src="https://s3-eu-west-1.amazonaws.com/static.euphoria.co.za/Libraries/AgentWorkspaceBundle/AgentWorkspaceBundle.min.js"></**script**>

</**head**>

<**body**>

<**div**>Status Box: <**span** id='stBox'></**span**></**div**>

<**div** id="incomingCall" style="display:none">Incoming Call</**div**>

</**br**>

</**br**>

<**button** onclick='Login()'>Login 500</**button**>

<**button** onclick='AgentWorkspace.LogOff();'>Logout</**button**>

<**button** onclick='AgentWorkspace.Pause("Tea");'>Pause</**button**>

<**button** onclick='AgentWorkspace.UnPause();'>UnPause</**button**>

<**button** onclick='Phone.Dial("0844929674")'>Dial 0844929674</**button**>

<**button** onclick='Phone.AnswerCall()'>Answer</**button**>

<**button** onclick='Phone.HangUpCall()'>End</**button**>

<**button** onclick='Phone.CancelCall()'>Cancel</**button**>

<**button** onclick='Phone.HoldCall()'>Hold</**button**>

<**button** onclick='Phone.UnHoldCall()'>Unhold</**button**>

<**button** onclick='Phone.MuteCall()'>Mute</**button**>

<**button** onclick='Phone.UnMuteCall()'>Unmute</**button**>

<**button** onclick='Phone.ChangeCallVolume(0)'>0 Volume</**button**>

<**button** onclick='Phone.ChangeCallVolume(1)'>100 Volume</**button**>

<**button** onclick='Phone.BlindTransfer("100") '>Transfer</**button**>

<**button** onclick='attendedTransfer("0712486258")'>Attended Transfer</**button**> *<!-- this function is custom function created at the bottom -->*

<**button** onclick='confirmTransfer()'>Confirm Transfer</**button**> *<!-- this function is custom function created at the bottom -->*

<**button** onclick='cancelTransfer()'>Cancel Transfer</**button**> *<!-- this function is custom function created at the bottom -->*

<**button** onclick='SubmitDisposition("Success", "Test"); '>Submit Dispo</**button**>

<**button** onclick='AgentWorkspace.SkipPreptime() '>Skip Prep</**button**>

</**br**>

</**br**>

<**button** onclick="Phone.DtmfSend('\*');">\*</**button**>

<**button** onclick="Phone.DtmfSend('#');">#</**button**>

<**button** onclick="Phone.DtmfSend('0');">0</**button**>

<**button** onclick="Phone.DtmfSend('1');">1</**button**>

<**button** onclick="Phone.DtmfSend('2');">2</**button**>

<**button** onclick="Phone.DtmfSend('3');">3</**button**>

<**button** onclick="Phone.DtmfSend('4');">4</**button**>

<**button** onclick="Phone.DtmfSend('5');">5</**button**>

<**button** onclick="Phone.DtmfSend('6');">6</**button**>

<**button** onclick="Phone.DtmfSend('7');">7</**button**>

<**button** onclick="Phone.DtmfSend('8');">8</**button**>

<**button** onclick="Phone.DtmfSend('9');">9</**button**>

</**br**>

</**br**>

<**h4**>Log: </**h4**>

<**div** id='statuses'></**div**>

<**script**>

function Login(sipUsername, sipPassword){

let authentication = {

Tenant: "<Your\_Tenant\_Name>",

ApiKey: "<Your\_API\_Key>",

LibraryKey: "<Your\_Library\_Key>"

}

let EuphoriaPhoneObj = {

WssServer: "<Your\_PBX\_Server\_URL>",

SipUsername: "<SIP\_Username>",

SipPassword: "<SIP\_Password>",

$StatusBox: $("#stBox")

}

Phone.Start(EuphoriaPhoneObj, authentication);

}

function SubmitDisposition(Disposition, Outcome){

var currentLine = Phone.**GetCurrentLine**();

if (currentLine) {

currentLine.crmTag = 'TestCrmTag';

currentLine.note = 'TestNote';

currentLine.**SubmitDisposition**(Disposition, Outcome);

}

else { console.log("No Current Line Found!"); }

}

**$(document).ready(function(){**

**Phone.Events.OnLoad(function () {**

***//triggers once the phone is ready to use***

**LogAction("Phone Loaded");**

**});**

**Phone.Events.OnRegister(function () {**

***//triggers on phone registration***

**LogAction("Phone Registered");**

**});**

**Phone.Events.OnUnregister(function () {**

***//triggers on phone deregistration***

**});**

**Phone.Events.OnRinging(function () {**

***//triggers on inbound ringing***

**LogAction("Ringing");**

**});**

**Phone.Events.OnDialError(function () {**

***// triggers if there was a dialing error***

**});**

**Phone.Events.OnCallAccept(function (callLine) {**

***// triggers on call answer (In/Out)***

**$("#incomingCall").hide();**

**LogAction("Call Accepted", callLine.number);**

**});**

**Phone.Events.OnDial(function (callLine) {**

***// triggers before dialing the number***

**LogAction("Dialled", callLine.number);**

**});**

**Phone.Events.OnAfterDial(function (callLine) {**

***// triggers after dialing the number***

**});**

**Phone.Events.OnCallTermination(function (callLine) {**

***// triggers once the call is terminated***

**$("#incomingCall").hide();**

**LogAction("Call Terminated", callLine.number);**

**if(callLine.dispoEnabled) LogAction("Waiting for disposition", callLine.number);**

**});**

**Phone.Events.OnIncomingCall(function (callLine) {**

***// triggers on incoming call***

**$("#incomingCall").show();**

**});**

**Phone.Events.OnCallFailure(function (callLine) {**

***// triggers if dialing failed***

**});**

**Phone.Events.OnHold(function (callLine) {**

***// triggers once the call is placed on hold***

**});**

**Phone.Events.OnUnHold(function (callLine) {**

***// triggers once the call is active after placed on hold***

**});**

**Phone.Events.OnMute(function (callLine) {**

***// triggers on call mute***

**});**

**Phone.Events.OnUnMute(function (callLine) {**

***// triggers on call unmute***

**});**

**Phone.Events.OnVolumeChange(function (callLine) {**

***// triggers on call volume change***

**});**

**Phone.Events.OnCallTransfer(function (callLine) {**

***// triggers after call transfer***

**});**

**AgentWorkspace.Events.OnDialerLeadAvailable(function(lead){**

***// number: num, // PD***

***// numbers: contactNumbers, // PR***

***// crmTag: crmTag,***

***// campaignName: campaignName,***

***// campaignType: campaignType,***

***// notes: notes,***

***// address: address,***

***// language: language,***

***// title: title,***

***// name: fullName,***

***// email: email,***

**LogAction("Lead Found", lead.name +" "+ lead.numbers[0].number);**

**});**

**});**

function LogAction(name, msg){

let datetime = new Date();

let time = datetime.getHours() + ":" + datetime.getMinutes() + ":" + datetime.getSeconds() ;

let msgToShow = time + " >> " + name;

if(msg) msgToShow += ": " + msg;

$("#statuses").append("<div>"+ msgToShow +"</div>");

}

var attendedTransferObject = null;

function attendedTransfer(number){

attendedTransferObject = Phone.AttendedTransfer(number);

}

function confirmTransfer(){

if(!attendedTransferObject) return;

attendedTransferObject.Confirm();

}

function cancelTransfer(){

if(!attendedTransferObject) return;

attendedTransferObject.Cancel();

}

</**script**>

</**body**>

</**html**>