Click-to-Call Cloud Service

Version 2016.05

Click-to-Call Setup API Specification

Click-to-Call Setup API Specification

Product version: 2016.05
Release date: 05-13-16
Document identifier: C2CCallSetupAPI1605131236

Table of Contents

1 Introduction

The Click-to-Call Cloud Service Call Setup API provides a RESTful web service to enable users to:

- Initiate a call
- Get the current status of a call
- End a call.

The API can be used by either visitor#facing browsers or customer servers.

This API Specification applies to version 2.0 of the Call Setup API.

How to Use This Document

This is a guide to show which chapters provide what information relating to the Click-to-Call Cloud Service Call Setup API.

Chapter 2 – *Using Call Setup API* (page 3) describes how to use the functionality provided by Call Setup API.

Chapter 3 – *Same Origin Policy* (page 21) provides information about how the Call Setup API is impacted by the Same Origin Policy and how to use Cross Origin Resource Sharing (CORS) or JSON with Padding (JSONP) with the Call Setup API.

How to Contact Oracle Support

If you have any questions or concerns regarding Click-to-Call Cloud Service you can contact Oracle via the My Oracle Support website. The website is available at https://support.oracle.com.

2 Using Call Setup API

This chapter provides detail about how to initiate a call, get the current status of a call and end a call using the Call Setup API.

API Interface Summary

The Call Setup API is designed as a secure RESTful web service. As such, it conforms to many of the guidelines for RESTful services, including:

- Using HTTP methods explicitly.
- Stateless design.
- Exposes directory structure#like URIs.
- Support for JSON for data transfer.

API URI

The URI for the Call Setup API starts with `/call/<version>`, where <version> denotes the version of the API. For the first official version of the API, <version> is 2.0, so the Call Setup API URI starts with `/call/2.0`.

Other path elements and parameters are required according to the action requested. In accordance with RESTful guidelines:

- Mandatory resource identification information should be passed as elements of the URI path.
- Optional information should be passed as request parameters.

API Server

All API requests must be directed to *api.atgsvcs.com*.

API Protocol

Customers wishing to use the Call Setup API must use HTTPS as the application protocol to securely communicate with the API server. The unsecure HTTP protocol is not supported.

API Actions

There are three main actions supported in the current version of the Call Setup API:

Action	HTTP Method
Start a call	POST
Get the current status of a call	GET
End a call	DELETE

Feedback from an API Request

The Call Setup API makes use of HTTP status codes to denote the outcome of the request. Please refer to *Appendix A: Use of HTTP Status Codes* (page 27) for further details.

Data Representations

JSON is the only format supported for request and response data.

Start a Call

This section provides instructions on how to start a call using the Call Setup API.

Start Call Request

To initiate a call between a visitor and a contact center, you need to issue a POST request to `/call/2.0/<contactId>/<userPhone>` where <contactId> refers to the call template that contains the call setup instructions, and <userPhone> contains the user's phone number in E.164 format.

The Call Templates section of this document provides more information about call templates.

The complete list of parameters supported by the Call Setup API for a start call request is:

Name	Usage	Passed As	Notes
contactId	Required	Path element	Identifies the call template definition that contains call setup instructions. Refer to the Call Templates (page 5) section below for further information about this request parameter.
userPhone	Required	Path element	Defines the end point to reach the user on. Must be a phone number in E.164 format.

Name	Usage	Passed As	Notes
delay	Optional	JSON attribute in request body	Indicates how many seconds to delay the start of the call by. The default value is 0 seconds.
userExtension	Optional	JSON attribute in request body	Indicates the extension on which the user can be reached. DTMF tones will be used to dial the extension number. The only characters permitted are numeric digits (0#9), period (.), comma (,), asterisk (*) and hash (#), where comma (,) represents a 2 second delay, and period (.) silences audio messages during the auto navigation.
visitorContext	Optional	JSON child object	Provides contextual information about the visitor and the call request. Refer to the Visitor Context (page 6) section below for information about what attributes this object contains.
authorizeUrl	Optional	JSON attribute in request body	Provides a URL that is used to authorize the start call request and to optionally override some attributes of the call.
authEncrypted	Optional	JSON attribute in request body	Indicates whether the URL passed in the 'authorizeUrl' parameter has been encrypted. Set to *true* if the URL is encrypted, or *false* if it is not. The default value is *false*.
callAgentFirst	Optional	JSON attribute in request body	Control whether the call leg ordering is reversed so that the agent is called before the user. Allowable values are 'true' and 'false'. The default value is 'false'. To turn on reverse call flow, set this to 'true'. **Note**: There is an account setting in Webcare which also must be turned on in order to permit reverse call flow calls.
when	Optional	JSON attribute in request body	Specifies the scheduled start time of the call. The value specified must be a date/time in the future. The format of the scheduled start time should be in RFC 2822 date/time format.

Call Templates

The mandatory *contactId* path element identifies a call template that contains setup information for the call such as:

- The customer account.

- The phone number to call for the agent call leg, including optional extension number.
- Optional audio to be played on one or both call legs.
- Optional ANI overrides to be used on one or both call legs.
- Pre#call authorization checks to be performed.

In legacy Click#to#Call, the invite links defined using the Webcare LinkBuilder interface were the only call templates supported. Version 1.0 of the Call Setup API supported LinkBuilder#based invite links are the main type of call template supported. Version 2.0 of the Call Setup API does not support LinkBuilder. Instead, version 2.0 supports a new form of call template.

Visitor Context

The optional *visitorContext* attribute (JSON object) that can be passed in the request data can contain the following contextual information about the visitor and the call request:

Attribute Name	Usage	Notes
visitorId	Optional	Identifies the visitor.
referrerUrl	Optional	URL of the page from which the call initiated.
language	Optional	Two#digit ISO code that identifies the language to be used for the call request.
<optionalDataFieldName>	Optional	Specifies value for the optional data field identified using either its account#specific label or the internal varN field name, where N is 1 to 10. Up to 10 optional data fields can be specified in this way. Refer to *Appendix B: Handling Optional Data* (page 29) for further information.

The Call Setup API also captures the caller's IP address from the HTTP request.

Start Call Response

Response for Successful Requests

If the start call request was successfully completed, then a 200 OK response is returned. The response body contains the following entities (JSON object members):

Entity Name	Data Type	String
sessionId	String	Unique ID for the call

Response for Unsuccessful Requests

If the request failed then an HTTP 4XX error status code is returned. The body of the response may provide details of why the request failed. HTTP error status codes specific to failed Call Status requests are as follows:

Error Condition	HTTP Status Code Used
Use of an invalid HTTP method for the action	405 Method Not Allowed
Invalid URI path	404 Not Found
API version (mandatory) is invalid	404 Not Found
Contact ID (mandatory) s invalid	404 Not Found
Call template identified by Contact ID cannot be found	404 Not Found
Visitor phone number (mandatory) is invalid	404 Not Found
Call delay parameter (optional) is invalid	400 Bad request
User extension parameter (optional) is invalid	400 Bad Request
Pre#call authorization checks fail	403 Forbidden
Start call attempt failed	404 Not Found

A full list of HTTP status codes used by the API can be found in *Appendix A: Use of HTTP Status Codes* (page 27).

Start Call Examples

Each of these examples shows the HTTP request and the response from the Call Setup API in each of the scenarios.

Example 1: Successful Call Using Basic Parameters

To setup a call between the user reachable on 12026345789 and the endpoint identified by the call template with the ID of 367901, issue the following request:

HTTP Request	**POST api.atgsvcs.com/call/2.0/367901/12026345789**
Request Body	Empty

The response received should be:

Response Status	**200 OK**
Response Content Type	application/json
Response Body	{ “sessionId”:”1TfU02Xfme2GMuzfUh0rkdgvz4” }

Example 2: Invalid request path

A request to a version of the Call Setup API that does not exist would fail. For example:

HTTP Request	**POST api.atgsvcs.com/call/2.0/367901/12026345789**
Request Body	Empty

The response received should be:

Response Status	404 Not Found

Example 3: Use of a Method other than POST

If you use PUT on the standard interface to try and start a call, the request will fail. For example:

HTTP Request	**PUT api.atgsvcs.com/call/2.0/367901/12026345789**
Request Body	Empty

The response received should be:

Response Status	405 Method Not Allowed

Example 4: Invalid contact ID

If the request passes a contact ID value that is either invalid or for which a call template does not exist, the request will fail. For example, assuming there is no call template with an ID of ABC123:

HTTP Request	POST api.atgsvcs.com/call/2.0/ABC123/12026345789
Request Body	Empty

The response received should be:

Response Status	404 Not Found

Example 5: Invalid visitor telephone number

If an invalid phone number is passed, then the request will fail. For example:

HTTP Request	POST api.atgsvcs.com/call/2.0/367901/1202
Request Body	Empty

The response received should be:

Response Status	404 Bad Request

Example 6: Pre-call authorization checks failed

If the pre#call authorization checks failed, for example because the visitor number was blacklisted, then the start call request will fail. For example, if number 12056545689 was blacklisted, then the following request:

HTTP Request	POST api.atgsvcs.com/call/2.0/367901/12056545689
Request Body	Empty

would return the following response:

Response Status	403 Forbidden

Example 7: Additional parameters

Consider a call with a number of additional parameters set, including two optional data fields with labels of CustRef and CartTotal. The Start Call request could look like this:

HTTP Request	**POST api.atgsvcs.com/call/2.0/367901/12026345789**
Request Content Type	application/json

Request Body	{
	“delay”:”60”,
	“userExtension”:”,123”,
	“visitorContext”:
	{
	“CustRef”:”ARC1235987”,
	“CartTotal”:”55.60”
	}
	}

The response received should be:

Response Status	**200 OK**
Response Content Type	application/json
Response Body	{ “sessionId”:”1TfU02Xfme2GMuzfUh0rkdgvz4” }

Example 8: Scheduling a call for a later date

Consider a request to schedule a call to be started at a later date (e.g. 12.30pm Eastern Time on Monday 25th July 2016) where the agent leg is to be dialled first. The minimum information to be sent in the HTTP request to the API should be as follows:

HTTP Request	**POST api.atgsvcs.com/call/2.0/367901/12026345789**
Request Content Type	application/json

HTTP Request	POST api.atgsvcs.com/call/2.0/367901/12026345789

Request Body	{
	“callAgentFirst”:”true”,
	“when”:”Mon, 25 July 2016 12:30 -0400”
	}

The response received should be:

Response Status	200 OK
Response Content Type	application/json
Response Body	{ “sessionId”:”1TfU02Xfme2GMuzfUh0rkdgvz4” }

If the date specified is in the past, the response received should be:

Response Status	400 Bad Request
Response Content Type	text/plain
Response Body	Start time is in the past: Mon, 27 July 2015 12:30 -0400

Get the current status of a call

This section provides instruction on how to get the current status of a call using the Call Setup API.

Get Call Status Request

To get the current status of a call, you need to issue a GET request to `/call/2.0/<sessionId>/status` where <*sessionId* is the session ID that uniquely identifies the call. This is the value returned in the *sessionId* attribute of the response for the originating Start Call request.

There are no additional parameters required to get the status of a call.

Get Call Status Response

Response for successful requests

If the request is successful, then a 200 OK response will be received. The response body will contain the following entities as JSON attributes:

Entity Name	Data Type	Description
sessionId	String	The session ID of the call. This is the same as the session ID that was passed in the status request.
callState	String	Code to denote the overall status of the call. The values for this attribute are *unknown*, *scheduled*, *initiated*, *connected*, and *disconnected*. Please refer to the Call States (page 13) section for more information on the call states.
time	Integer	Indicates the time, in seconds since the Epoch (January 1 1970 00:00:00 UTC), that the specified state happened. This attribute is not returned for unknown call states.
scheduledTime	Integer	Indicates the time, in seconds since the Epoch (January 1 1970 00:00:00 UTC), at which the call is scheduled to start. This attribute is only returned for scheduled call state.
more	String	Indicates if there may be further changes in the call state. Values are: no: This is used if the call has ended. yes: This is used if the call is in progress. maybe: This is used if the call state is unknown, or if the call state has not changed since the last status request.
hasChanged	String	Indicates if the status of the call has changed since the last time its status was requested.

Response for unsuccessful requests

If the request failed then an HTTP 4XX error status code will be returned. The body of the response may provide details of why the request failed. HTTP error status codes specific to failed Get Call Status requests are as follows:

Error Condition	HTTP Status Code Used
Invalid URI path	404 Bad Request

Error Condition	HTTP Status Code Used
Invalid request parameters	400 Bad Request
Use of an invalid HTTP method	405 Method Not Allowed
Call session does not exist	404 Not Found

A full list of HTTP status codes used by the API can be found in *Appendix A: Use of HTTP Status Codes* (page 27).

Call States

The following table lists the possible call states that can be returned to the caller, along with their allowable 'more' and 'time' response settings:

callState	Description	time	more
Unknown	The current status of the call is unknown.	Omitted	Maybe
Scheduled	The call has been scheduled to start at a later time.	Time in seconds since Epoch that the scheduled call was requested.	No
Initiated	The call has been initiated.	Time in seconds since Epoch that call was initiated.	Yes
Connected	The call (or call leg) has been connected.	Time in seconds since Epoch that call was connected.	Yes
Disconnected	The call (or call leg) has been disconnected.	Time in seconds since Epoch that call was disconnected.	No

Long Polling Mechanism

The Call Setup API makes use of a long#polling mechanism for the Get Call Status request. When a request is made, the server#side API component will only return a response when either the call status has changed since the last request, or the long poll timeout period for the request has elapsed.

If the call status has changed since the last request, the Call Setup API returns the following attribute values:

Response Attribute	Value
callState	The new state of the call.
more	The time that the new state took effect. Omitted for *unknown* state.

Response Attribute	Value
time	This is set according to the new state: *maybe* if the new state is `unknown` *no* if the new state is `scheduled` or `disconnected` *yes* if the new state is `initiated` or `connected`
hasChanged	This is set to *yes*.

If the long poll timeout period is reached, the Call Setup API returns the following attribute values:

Response Attribute	Value
callState	The new state of the call.
more	The time that the new state took effect. Omitted for *unknown* state.
scheduledTime	If the current state is *scheduled* then this will denote the scheduled start time of the call. Omitted for all other call states.
time	This is set according to the new state: *maybe* if the new state is `unknown` *no* if the new state is `disconnected` *yes* if the new state is `initiated` or `connected`
hasChanged	This is set to *no*.

The default timeout period for each request is 30 seconds. This can be configured using the 'waitperiod' request parameter which specifies the wait period in seconds. The maximum wait period that is permitted is 120 seconds (2 minutes).

This mechanism helps to reduce the number of requests that must be made to track the current status of a call.

Session Token Expiry

When a call session has ended more than 15 minutes prior to the status request, the Call Setup API expires the session token. This means that if a request is made to get the status of the call session after that time, the Call Setup API returns an *HTTP 404 Not Found* response.

Get Call Status Examples

Each of these examples shows the HTTP request and the response from the Call Setup API in each of the scenarios.

Example 1: Get the status of a call that is scheduled to start in the future

HTTP Request	GET /call/2.0/NND4QvEgFlltU0ObJtGbgGRfO4X/status

The response should be:

Response Status	**200 OK**
Response Content Type	application/json
Response Body	{“sessionid”:”NND4QvEgFlltU0ObJtGbgGRfO4X”,”callState”:”scheduled”, "time":"2013-07-25 4:35:50","scheduledTime":"2013-07-29 16:30:00",“more”:”no”,”hasChanged”:”yes”}

Example 2: Get the status of a call that is in the process of starting

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	**200 OK**
Response Content Type	application/json
Response Body	{"sessionid":"NND4QvEgFlltU0ObJtGbgGRfO4X","callState":"unknown", "more":"maybe","hasChanged":"yes"}

Example 3: Get the status of a call that has now started

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	200 OK
Response Content Type	application/json
Response Body	{“sessionid”:”1TfU02Xfme2GMuzfUh0rkdgvz4”,”callState”:”initiated”,”time”:” 1343129570”,”more”:”yes”, “hasChanged”:”yes”}

Example 4: Get the status of a call that has now connected

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	200 OK
Response Content Type	application/json
Response Body	{“sessionid”:”1TfU02Xfme2GMuzfUh0rkdgvz4”,”callState”:”connected”,”time”:” 1343129590”,”more”:”yes”, “hasChanged”:”yes”}

Example 5: Get the status of a call where the current state is still connected

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	200 OK
Response Content Type	application/json
Response Body	{“sessionid”:”1TfU02Xfme2GMuzfUh0rkdgvz4”,”callState”:”connected”,”time”:” 1343129590”,”more”:”yes”, “hasChanged”:”no”}

Example 6: Get the status of a call that has now ended

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	**200 OK**
Response Content Type	application/json
Response Body	{“sessionid”:”1TfU02Xfme2GMuzfUh0rkdgvz4”,”callState”:”disconnected”,”time”:”1343129630”, “hasChanged”:”yes”}

Example 7: Get the status of a call whose session token has expired

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	**404 Not Found**
Response Content Type	text/plain
Response Body	Call session no longer exists because it has ended

Example 8: Request with an invalid URI path

In this example, the Get Call Status request is missing part of the URI.

HTTP Request	GET /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4

The response should be:

Response Status	404 Not Found

Example 9: Invalid HTTP method

In this example, the wrong HTTP method is used in the Get Call Status request.

HTTP Request	PUT /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	405 Method Not Allowed

End a Call

This section provides instruction on how to end a call using the Call Setup API.

End Call Request

To end a call, you need to issue a DELETE request to `/call/2.0/<sessionId>` where <sessionId> is the session ID that uniquely identifies the call. This is the value returned in the sessionId attribute of the response to the originating start call request.

No additional parameters are required for the DELETE call.

End Call Response

Response for successful requests

If the request is successful a *200 OK* response should be received. The response body contains the following entity as a JSON attribute:

Entity Name	Data Type	Description
callState	String	This holds the value *ended* to denote that the call has ended.

Response for unsuccessful requests

If the request failed then an HTTP 4XX error status code should be returned. The body of the response may provide details of why the request failed.

Error codes specific to failed End Call requests are as follows:

Error Condition	HTTP Status Code Used
Invalid URI path	404 Not Found
Use of an invalid HTTP method	405 Method Not Allowed
Call session does not exist	404 Not Found
The call is already ending	403 Forbidden

A full list of HTTP status codes used by the Call Setup API can be found in Appendix 1: Use of HTTP Status Codes.

End Call Examples

Each of these examples shows the HTTP request and the response from the Call Setup API in each of the scenarios.

Example 1: End a call that is in progress

HTTP Request	DELETE /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4

The response should be:

Response Status	**200 OK**
Response Body	{“callState”:”ended”}

Example 2: End a call that is already ending or has already ended

HTTP Request	DELETE /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4

The response should be:

Response Status	**403 Forbidden**
Response Body	The call has already ended.

Example 3: End a call that does not exist

HTTP Request	DELETE /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4

The response should be:

Response Status	**404 Not Found**
Response Body	Attempt to end the call failed.

Example 4: Request with an invalid URI path

In this example, the Get Call Status request is missing part of the URI.

HTTP Request	DELETE /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4

The response should be:

Response Status	404 Bad Request

Example 5: Invalid HTTP method

This table is described in surrounding text.

In this example, the wrong HTTP method is used in the Get Call Status request.

HTTP Request	PUT /call/2.0/1TfU02Xfme2GMuzfUh0rkdgvz4/status

The response should be:

Response Status	405 Method Not Allowed

3 Same Origin Policy

This chapter provides more information about how the Call Setup API is impacted by the Same Origin Policy and how to use Cross Origin Resource Sharing (CORS) or JSON with Padding (JSONP) with the Call Setup API.

Impact of Same Origin Policy on the Call Setup API

Cross#site access from a visitor browser is usually denied by the same origin policy implemented by the browser. Two resources are considered to be of the same origin if they share the same domain name, port number and protocol. However the HTML <script> element is able to perform content retrieval from foreign origins.

For request to the Call Setup API from a visitor browser, the origin of the Call Setup API will usually differ from that of the customer site that the visitor is visiting. Therefore we need to provide a workaround to the same origin policy.

JSONP (JSON with Padding) and CORS (Cross#Origin Resource Sharing) provide two ways in which we can provide a workaround to the same origin policy. CORS is a more modern alternative to JSONP and is supported by more modern browsers. This will be supported by the Call API.

The Call API is expected to support any browser version with a market share of 2% or greater. This includes a number of browser versions that do not support CORS. This currently includes:

Level of support for CORS	Browser Version
None	IE7 and earlier Firefox 3.0 and earlier Safari 3.x and earlier Opera 11 and earlier
Partial	IE8 and IE9

For browsers that do not support CORS, JSONP is supported as a fallback. JSONP does not support the POST or DELETE methods, so GET alternatives for the Start Call and End Call API methods have to be provided.

Once all supported browser versions support CORS, support for JSONP will be deprecated.

Overview of CORS

Cross#Origin Resource Sharing (CORS), which has been proposed by the Web Applications Working Group, provides a way for web servers to support cross#site access controls, which enable secure cross-site data transfers.

The Cross#Origin Resource Sharing standard works by adding new HTTP headers that allow servers to describe the set of origins that are permitted to read that information using a web browser.

Additionally, for HTTP request methods that can cause side#effects on user data (in particular, for HTTP methods other than GET, or for POST usage with certain MIME types), the specification mandates that browsers pre#flight the request, soliciting supported methods from the server with an HTTP OPTIONS request header, and then, upon "approval" from the server, sending the actual request with the actual HTTP request method. Servers can also notify clients whether "credentials" (including Cookies and HTTP Authentication data) should be sent with requests.

The CORS#specific headers that can be used in cross#origin and pre#flight requests can be found in the Syntax section of the W3C Working Draft for CORS.

In order to support cross#domain request, the Call Setup API has supported CORS since version 1.0.

Overview of JSONP

JSONP provides a technique for accessing data from a different domain. With JSONP, the client provides a call back function which will process the JSON payload received from the server.

JSONP Interface to the Call Setup API

The Call Setup API supports JSONP as follows:

- The HTTP GET method is supported for Start Call, Get Call Status and End Call methods.
- A callback request parameter can be used to specify the name of the callback function.

For both successful and unsuccessful requests, the response body will contain a JavaScript function call where the name of the function is specified with the callback request parameter. For example, if the function name specified in the callback request parameter is processResponse then the response body should be:

```
processResponse({"param1":"value1", "param2":"value2"});
```

where *param1*, *value1*, *param2*, and *value2* are the JSON attribute names and values for the response.

In the case of successful requests, the expected JSON attributes for the response are passed as parameters of the function call.

In the case of unsuccessful requests, there are two JSON attributes passed in the function call. These are:

Attribute Name	Data Type	Description
statuscode	Number	Holds the HTTP status code for the response. For example, 404, 403, 400.
reason	String	Reason for the unsuccessful request.

Further information specific to the method is in each of the following sections.

Start Call Request using JSONP

To setup and start a call using the JSONP interface, you need to issue a GET request to `/call/2.0/<contactId>/<userPhone>/start?callback=<callbackFunction><otherCallSetupParams>` where:

- <contactId> identifies the call template that contains call setup instructions. Refer to the Call Templates (page 5) section of this document for more information about call templates.
- <userPhone> contains the user's telephone number in E.164 format.
- <callbackFunction> specifies the name of the callback function to which the server-side JSON response should be passed.
- <otherCallSetupParams> lists other optional call setup parameters, each one preceded by an ampersand (&).

  The optional request parameters that may be passed are:

Parameter Name	Description
delay	Number of seconds to delay the start of the call by. The default value is 0.
userExtension	Indicates the extension that the user can be reached on. DTMF tones will be used to dial the extension number. The only characters permitted are numeric digits (0#9), period (.), comma (,), asterisk (*) and hash (#), where comma (,) represents a 2 second delay, and period (.) silences audio messages during the auto navigation.
visitorId	Identifies the visitor.
referrerUrl	URL of the page from which the call originated.
language	Two#digit ISO code identifying the visitor's language.
<optionalDataFieldName>	Specifies value for the optional data field identified using either its account#specific label or the internal varN field name. Up to 10 optional data fields can be specified in this way. Refer to *Appendix B: Handling Optional Data* (page 29) for further information.
authorizeUrl	Provides a URL that is used to authorize the start call request and to optionally override some attributes of the call.

Parameter Name	Description
authEncrypted	Indicates whether the URL passed in the 'authorizeUrl' parameter has been encrypted. Set to *true* if the URL is encrypted, or *false* if it is not. The default value is *false*.
confirmAcceptance	Controls whether the user is prompted to confirm acceptance of the call. Allowable values are 'true' and 'false'. The default value is 'false'. To turn on call confirmation acceptance, set this to 'true'.
callAgentFirst	Control whether the call leg ordering is reversed so that the agent is called before the user. Allowable values are 'true' and 'false'. The default value is 'false'. To turn reverse call flow, set this to 'true'.
when	Specifies the scheduled start time of the call. The value specified must be a date/time in the future. The format of the scheduled start time should be in RFC 2822 date/time format.

If the Start Call request is successful, then the response body should contain a JavaScript function call with the JSON response, which includes the session ID for the call, embedded in the function call. The name of function is specified in the callback request parameter.

For example, if the callback request parameter contains the name 'processStartCallResponse', then the response body is:

```
processStartCallResponse({"sessionId":"<sessionId>"});
```

where <sessionId> is the session ID for the new call.

If the Start Call request is unsuccessful, the response body is:

```
processStartCallResponse({"statuscode":"<statusCode>","reason":"<reason>"});
```

where <statusCode> contains the HTTP status code (i.e. 400, 403, 404) and <reason> contains the reason for the unsuccessful request.

Get Current Status Request using JSONP

To get the current status of a call using the JSONP interface, you need to issue a GET request to `/call/2.0/<sessionId>/status?callback=<callbackFunction>` where:

- <sessionId> is the session ID that uniquely identifies the call. This is the value returned in the *sessionId* attribute of the response for the originating start call request.
- <callbackFunction> specifies the name of the call back function to which the server#side JSON response is passed.

If the request is successful, then the response body should contain a JavaScript function call with the JSONresponse, which contains call status information for the call, embedded in the function call. The name of the function is specified in the callback request parameter.

For example, if the callback request parameter contains the name 'processCallStatusResponse', then the response body is as follows (for a connected call):

```
processCallStatusResponse({"sessionid":"<sessionId>","callState":"connected","time":"<connectedTime
 "hasChanged":"yes"});
```

where <sessionId> is the session ID for the new call and <connectedTime> is the time that the call was connected.

If the request is unsuccessful, the response body is:

```
processCallStatusResponse({"statuscode":"<statusCode>","reason":"<reason>"});
```

where <statusCode> contains the HTTP status code (i.e. 400, 403, 404) and <reason> contains the reason for the unsuccessful request.

End Call Request using JSONP

To end a call using the JSONP interface, you need to issue a GET request to `/call/2.0/<sessionId>/end?callback=<callbackFunction>` where:

- <sessionId> is the session ID that uniquely identifies the call. This is the value returned in the *sessionId* attribute of the response for the originating start call request.
- <callbackFunction> specifies the name of the call back function to which the server#side JSON response is passed.

If the request is successful, then the response body should contain a JavaScript function call with the JSONresponse, which contains call status information for the call, embedded in the function call. The name of the function is specified in the callback request parameter.

For example, if the callback request parameter contains the name 'processEndCallResponse', then the response body is as follows (for a connected call):

```
processEndCallResponse({"callState":"ending"});
```

If the request is unsuccessful, the response body is:

```
processCallStatusResponse({"statuscode":"<statusCode>","reason":"<reason>"});
```

where <statusCode> contains the HTTP status code (i.e. 400, 403, 404) and <reason> contains the reason for the unsuccessful request.

4 Appendix A: Use of HTTP Status Codes

The following table lists the HTTP status codes that are used by the Call Setup API to denote the success or failure of a request made to the Call Setup API:

HTTP Status Code and Text	Usage
200 OK	The request was successful.
400 Bad Request	Invalid contextual information passed for the request. For example: Invalid call delay period. Invalid user extension.
403 Forbidden	The server understood the request but is refusing to fulfil it. For example: Pre#call authorization failed Unable to end a call that is already ending.
404 Not Found	Resource is not accessible. For example: Badly formed request URI. A resource identified in the URI is invalid or does not exist. For example: version, contact ID, phone number, session token Call could not be started. Call session does not exist (or has gone away).
405 Method Not Allowed	HTTP method is not supported for the request.

5 Appendix B: Handling Optional Data

The Click#to#Call product can support up to 10 optional data fields which are used to hold contextual information for a call. This can include information derived from the call template and also information gleaned from the visitor's session on the customer's site.

These 10 fields have system#wide internal names of var1 to var10. Each customer account can define their own names or labels for these fields.

Default information for some of the fields is provided automatically from the call template. This can be overridden by information derived from the visitor session. To pass information from the visitor session, you simply need to define the JSON variables and values accordingly.

The optional data fields can be identified using their customer#specific labels for these fields or using the system#wide internal names. For example, to set the value of the third optional data field to XYZ54321, where its customer#specific label is 'CustRef', and the default name is 'var3', then this can be done in either of the following two ways, using JSON notation:

- "var3":"XYZ54321"
- "CustRef"="XYZ54321"

6 Glossary

Term	Definition
ANI	Automatic Number Identification Identifies who is calling the recipient of the call. With Click#to#Call this feature is used to provide a more meaningful caller ID than an eStara#specific caller number.
CORS	Cross Origin Resource Sharing defines ways for a web server to allow its resources to be accessed by a web page from a different domain.
Customer, Customer Account, or Account	This is an organization that has subscribed to one or more Live Help services such as Call or Chat. Such an organization will make use of one or more of our rules engines to present call and chat invites on their consumer#facing websites.
E.164	E.164 is an ITU#T recommendation that defines the international public telecommunication numbering plan used in the PSTN and some other data networks. It also defines the format of telephone numbers. E.164 numbers can have a maximum of fifteen digits and are usually written with a + prefix. To actually dial such numbers from a normal fixed line phone, the appropriate international call prefix must be used.
End User, Visitor, or Consumer	This represents the end user who visits the web site(s) of the customer account and optionally clicks on a call or chat invite when he or she requires assistance.
HTTP	Hypertext Transfer Protocol Application protocol that is the foundation for data communication for the World Wide Web.
JSON	JavaScript Object Notation A text#based open standard for data interchange that is more lightweight than XML for serializing and transmitting data between a server and a web application.

Term	Definition
JSONP	JSON with Padding This is a complement to the base JSON data format. It provides a way of requesting data from a server in a different domain, which is usually prohibited by web browsers because of the same origin policy. It can be used instead of CORS for web browsers that don't support CORS.
Long Polling	A variation of the traditional polling technique that allows emulation of an information push from a server to a client. With long polling, the client requests information from the server in a similar way to a normal poll. However, if the server does not have any information available for the client, instead of sending an empty response, the server holds the request and waits for some information to be available. Once the information becomes available, or after a suitable timeout, a complete response is sent to the client.
REST	Representational State Transfer This is a style of software architecture for distributed systems. It has emerged as a key web service design model that has increasingly displaced other design models such as SOAP and WSDL.
RESTful Web Service	A web service implemented using HTTP and the principles of REST.
SIP	Session Initiation Protocol IP network protocol mainly used for VoIP (Voice Over IP) telephony.