August 28, 2020

Customer Engagement API

User Manual

NUANCE

Table of Contents

Overview

The Customer Engagement API (CEAPI) is used to power third-party messaging channels and native mobile applications on the Nuance Digital Engagement Platform (NDEP).

The Nuance CEAPI allows any application that supports HTTP to utilize the Nuance chat system using a REST API. The CEAPI supports communication transmissions between a customer and an agent for the purpose of supporting chat. Services that are part of a standard web experience but not part of the core communication between customer and agent are not supported. Here is a list of items and communications that are not supported.

- Font size should be implemented by the UI container
- Print functionality should be implemented by the UI container (possible extension in the future)
- Pre-chat should be implemented by the UI container
- The “thank you” image should be implemented by the UI container.
- To avoid losing an engagement, every next message should be sent in no less than 60 seconds. Otherwise the engagement will close due to timeout.

API Token Authentication

Nuance Data Collection API (DCAPI) now supports the OAuth2-based authorization framework for application-level tokens. OAuth2 provides authorization based on the Client Credentials Grant Flow of the OAuth2 specification.

Here is a list of the steps required to access the API using OAuth2-based authorization.

- Client registers their application with Nuance.
- Client application requests the access token using the Authorization Server.
- Client application uses the access token in every API request.

Registering the Application

Clients may request application registration with Nuance by contacting their Client Services Manager (CSM). The CSM will register your application with the Nuance System Team.

The **clientId** is used by the service API to identify your application and to build login URLs.

The **clientSecret** is used to authenticate the identity of your application. It must be kept private between the application and the API.

Note. The client is responsible for storing the clientID and the clientSecret securely. If this information is compromised, you will have to request a lock on the account and request a new registration with your CSM.

Obtaining an Access Token

The client application must connect to the Auth server using the clientID and clientSecret provided by Nuance. The clientId and clientSecret must be formatted properly for transmission. Formatting requires you to concatenate the clientId, a colon ":" character, and the clientSecret into a single string as follows:

```
clientId:clientSecret
```

The resulting string should be Base-64 encoded and inserted into the Authorization Header in the following format:

```
Authorization: Basic [Base-64 encoded string]
```

Example

```
Request URL: https://auth.touchcommerce.com/oauth-server/oauth/token
Response Format: JSON
```

Note. Domain names may vary for each client.

Sample Request for Obtaining an Access Token

```
POST https://auth.touchcommerce.com/oauth-server/oauth/token
```

Form Data

```
client_id:"ceapiId"
grant_type:"client_credentials"
```

Form Data Details

- **client_id** – The **clientId** provided by the CSM when the client application was registered with Nuance.
- **grant_type** – This value should be **client_credentials** for the CE-API. When Nuance adds for other APIs, this value may be different for the new APIs.

Request Header:

```
Host: auth.touchcommerce.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Authorization: Basic Y2VhcGlDbGllbnRJZDpjZWFwaUNsaWVudFNlY3JldA==
```

Sample Response from Nuance


```
{
access_token:
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsiY2VhcGkiXSwic2NvcGUiOlsicmVhZCIsIndyaXRlIl0sInNpdGVzIjpb
IjE1NCIsIjMwNiJdLCJleHAiOjE0ODIyNzMyOTMsImp0aSI6IjM1ZjNjYzVmLWEyMmUtNDc5Yi1iNTM4LWIxYTkzNjI3YTk1YiIsImNsaW
VudF9pZCI6ImNlYXBpQ2xpZW50SWQifQ.jU3SaGeHYkxNaRVdbZhJFcI6rbeKjXQqfnjccib4gsChs9Ed2ZpbjgONpOUNdwJFwwE3BT3h-
wDRiaSUqieczOXc058aI1yGuAwr9nDDy4ex5sP9kGsOMEkw3PJaIdYfqAKJW2BU9LJ1efvhPGNHzRv7ivOHVNHWacB-
XeaGVVjbXXx4AJrHDrP-C-
0fiK7AlCNuaYIGV47c9e3_479MGJpfEVcQVhhoZB6DlbVtrf3IFPLD456M5MoqC0kTGn2wJfmUSM77eCNDcC7cMdDokvb1sAaOmhLMaJku
GfmvkXT7v7Yqm0_LA4GsHcPsmN9J8Dk4oyuhUTJYunBsdr_RRQ",
token_type: "bearer",
expires_in: 59,
scope: "read write",
sites : [ "154", "306" ],
jti : "10dc19a9-48c4-4446-b141-d5179e5f21f1"
}
```


Response Details

- **Access token** – The access token must be included in all API data requests.
- **Token type** – The token type is “bearer” and must be included in all API data requests.
- **Expires in** – This value indicates the expiration time of the token in seconds. The client app should monitor this value and request a new token whenever necessary. If the app uses an expired token the server will return a code 401 “Access Token Expired” message.
- **Scope** – Scope indicates the access permissions for this token. If a disallowed action is performed, the server will return an error.
- **Sites** – Indicates the site IDs that are valid for this token. If the request tries to access a site that is not on the list, the server will return an error.
- **jti** – The ID for the access token that is used for tracking purposes. Clients can ignore this value.

Using the Access Token in an API Request

The access token must be present in every API data request. It must be in the authorization header in the following format:

```
Authorization: Bearer [access token]
```

Sample API Request with Bearer Token

```
POST  https://api.touchcommerce.com/engagementAPI/v2/customer/agentAvailability?businessUnitID=22&output=JSON&siteID=306
```

Note. Domain names may vary for each client.

Request header:

```
Host: api.touchcommerce.com

Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsiY2VhcGkiXSwic2NvcGUiOlsicmVhZCIsIndyaXRlIl0sInNpdGVzIjpbI
jE1NCIsIjMwNiJdLCJleHAiOjE0ODIyNzM5MDYsImp0aSI6IjEwZGMxOWE5LTQ4YzQtNDQ0Ni1iMTQxLWQ1MTc5ZTVmMjFmMSIsImNsaWV
udF9pZCI6ImNlYXBpQ2xpZW50SWQifQ.Ax348la0IXKY_iTIYgvKcJB2_KJBRJcN5Td0Nk5546jhpun1pw4XN81PBtE0U4V7mT-
F9GOPTJhKKlpd5NPe-
ugmiSImQhncHzy3jE0_L0vToXzcxBGDu4EBzrmnWV29zZfQ_bfz7XIrpk7xxuRXQnWkGTlnVn2RYhek9HDQN9psGfJEqyRAcgHc4Lw26Ux
UqC4sz8_80mdMr8eUe6rOVHmiYOd9bVpCX4yYYRg9_5AFpZG-
BzkugbMu2CY7IFR1VxbGsGdF1GIUdX_C7j03hfykLBjCmEsD_jhKDVKCWINqbmMEqg4LsVbKqB6qFMh1OuOYjrFrxWm_i-GNqNzUPg
```

Sample API Response from Nuance

```
200 OK
```

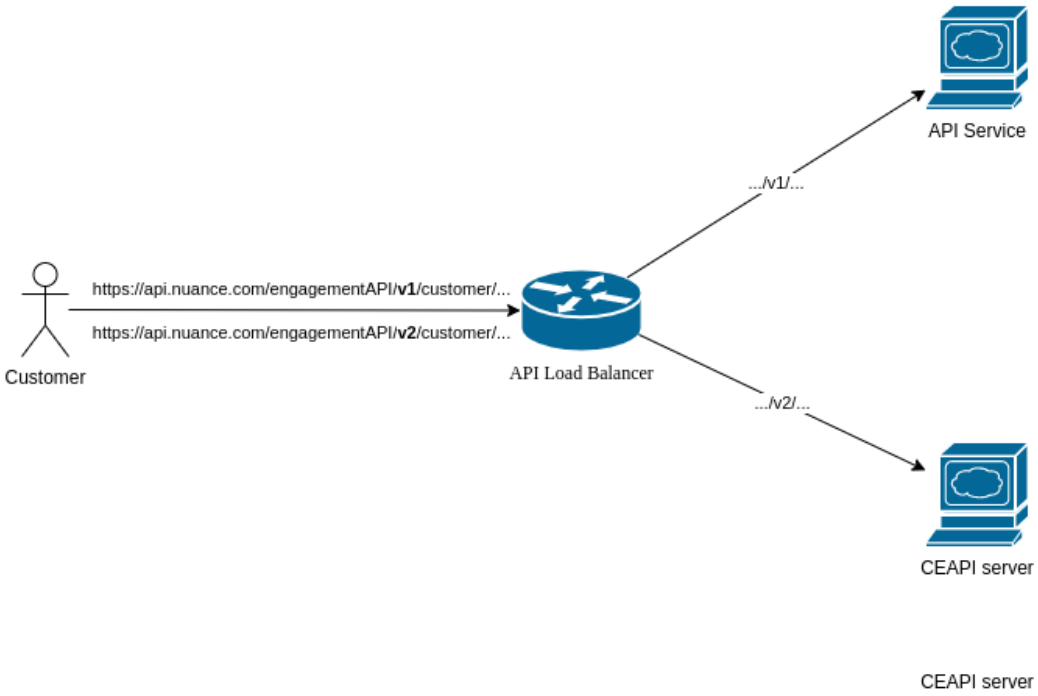

NUANCE

Load Balancer Settings

CEAPI v1 and v2 are handled by the same load balancer: API Load Balancer. The v1 requests are handled by the API service and v2 requests are handled by the CEAPI server. The client does not need to change the domain to reach the API service or the CEAPI servers. The logic to reach the API service or the CEAPI server is hidden in the load balancer settings:

The load balancer parses the URL and checks the version of the request.

- v1 requests are redirected to the API service
- v2 requests are redirected to the CEAPI server.



Activity

Post activity is used to send chat activity messages into the system. These messages will not be stored in the chat transcript.

URI	activity
HTTP Method	POST
Content type	application/x-www-form-urlencoded

Parameters

Name	Value	Type	Required
activityType	Activity type, currently available: customerStartTyping, customerStopTyping, customerMinimizeWindow, customerRestoreWindow, uploadFile.	string	yes
customerID	The customerID value returned by the Engagement Request response.	string	yes
engagementID	The engagementID value returned by the Engagement Request response.	string	yes
messageText	Custom text for activity. It is used only for uploadFile activity.	string	Yes for uploadFile only

POST Example of Sending a Message

POST `https://api.touchcommerce.com/engagementAPI/v2/customer/activity`

customerID=111&engagementID=111&activityType=customerStartTyping

Response Example

```
200 OK
```

Activity Types

Activity types customerStartTyping and customerStopTyping tell the agent when the customer starts or stops typing. Start typing should only be repeated if there was a stop typing request. The stop typing activity type should be sent when there is no user input for a timeout period (such as 4 seconds, for example.)

The following pseudo code sketches the algorithm for the client to send start and stop typing:

```
boolean isTyping = false;

Timer stopedTypingTimer = null;

// event handler for user typing event (onKeyPress, onChange or similar event)
function onUserTyping() {
  // reschedule the stoppedTyping timer to the new time = now + 4seconds
  cancelTimerIfNotNull(stoppedTypingTimer);
  stoppedTypingTimer = scheduleTimer(now + 4sec, onStoppedTyping);

  if (!isTyping) {
    POST activity?activityType=customerStartTyping
    isTyping = true;
  }
}

function onStoppedTyping() {
  isTyping = false;
  POST activity?activityType=customerStopTyping
}
```

Agent Availability

The **Agent Availability** call determines the current agent availability status.

Agent availability status depends on currently active engagements, total agent "slots" (sum of available slots across all agents in the relevant agent group) and queue threshold (configured per agent group). The primary value in the response is agent availability (true or false) which is based on the following algorithm:

$$(Queue\ Threshold) * (Total\ Agent\ Slots) - ((Total\ Active\ Engagements) + (Total\ Queued\ Engagements))$$

If this number is greater than 0, agent availability will be true. If it is less than or equal to 0 agent availability will be false.

Agent availability is typically used to decide whether to present an invitation to engage or to determine whether a program is within the hours of operation.

URI	agentAvailibility
HTTP Method	GET
Formats	XML or JSON

Parameters

Name	Value	Type	Required
agentAttributes	List of agent attributes used in the routing process. Format: attribute1,value1;attribute2,value2;attribute3,value3 Example: location,SD;location,AH;timezone,pst **Note:** Agent attributes are used to specify the attributes used for routing (sometimes called routing attributes). If set, this parameter will include the named attributes when checking availability or requesting an engagement.	string	No

Name	Value	Type	Required
	This call will look for an agent who possesses all of the listed skills.		
agentGroupID	Agent Group ID to check agent availability. **Note**: This parameter is only required for sites that have agent groups.	string	Yes
businessUnitID	Business Unit ID to check agent availability and HOP availability.	string	Yes
output	XML or JSON. Default is XML.	string	No
queueThreshold	Queue threshold to check agent availability. If this value is not specified it will take the Portal or default value.	number	No
siteID	Site ID to check agent availability	string	Yes

Response Values

Name	Value	Type	Description
availability	true or false	boolean	Based on request content and related program configuration/utilization, is there an agent or queue slot available at the time of the request
estimatedWaitTime	number	number	Estimated wait time in seconds.
inHOP	true or false	boolean	Hours of Operation is defined at the site or Agent Group level. This represents when a program is open or closed. True is returned if the program is open. False is returned if the program is closed.
queueDepth	number	number	Number of chats waiting in the queue.

Name	Value	Type	Description
status	online, busy, offline	string	Indicates whether there is an active agent with a chat slot available (online), there are agents logged in, but all their chat slots are full (busy), or all agents on a busy status (ex. “break) or are not logged in (offline). **Note**: The status will return “busy” if all agent slots are full, even when there are queue slots available.

Request Example

```
Get https:// api.touchcommerce.com/engagementAPI/v2/customer/agentAvailability

GET parameters: siteID=10003715@businessUnitID=1300508&agentGroupID=10004026
```

Note: The Business Unit ID parameter is required.

Response Example

```
200 OK

<?xml version="1.0" encoding="utf-8"?>
<message>
  <inHOP>true</inHOP>

<status>online</status><availability>true</availability><queueDepth>0</queueDepth><estimatedWaitTime>0</estimatedWaitTime></message></message>
```

Note: If all chat slots are taken up and the agent gets another chat request, the status will be *busy,* and availability will be *false*. An example is shown below.

```
<?xml version="1.0" encoding="UTF-8"?><message><inHOP>true</inHOP><status>busy</status><availability>False</availability><queueDepth>0</queueDepth><estimatedWaitTime>0</estimatedWaitTime></message>Assisted
```

The **Assisted** endpoint allows applications to indicate when the criteria has been met for assisting a customer using an engagement.

URI	event/assisted
HTTP Method	POST
Formats	JSON

Parameters

Name	Value	Type	Required
attributes	List of any modifiable name/value attributes that you want to associate to this customer, engagement or session	N/A	*No*
attributes:action	Action to be performed on this attribute	Must be one of ***append***, ***set*** or ***remove***. Default is ***append.***	*No*
attributes:attributeType	Type of attribute that we are setting or modifying	Must be one of ***engagementAttribute***, ***visitorAttribute*** or ***applicationAttribute***	*Yes*
attributes:externalCustomerID	Can be used only for ***visitor attributes***. If this ***visitor attribute*** can be used as a unique identifier for this user, please indicate it using a **yes** value in this field.	Must be either ***yes*** or ***no***, default is ***no***	*No*
attributes:name	Name of this attribute.	String	*Yes*
attributes:value	Value of this attribute. Can be multi-valued.	String	*Yes*

Name	Value	Type	Required
agentGroupID	Id of the current agent group is provided by *Nuance* to use on the *Nuance* platform. This parameter must be the same as the engagement had on the **/engagement** call. The default value is *agentGroupID* from the **/engagement** request. This parameter is used only to determine hours of operation in cases where both *agentGroupID* and *businessUnitID* are set next: get this data by *agentGroupID* value first, and after by *businessUnitID* parameter. If both business unit level and agent group level settings are configured, then only the agent group level settings are used. Hours of operation is always true. If this field is absent and *businessUnitID* is also not set.	Integer	*No*
businessRuleID	Business rule ID is used to tie this engagement to a business rule for reporting purposes. This field must have the same value as the engagement had on ***/engagement*** call. Default value is *businessRuleID* from ***/engagement*** request	String, possible values are: [a-z], [A-Z], [0-9], [-], [_]	*Yes*
businessUnitID	Id of current business unit provided by *Nuance* to use *Nuance* platform. This parameter must be the same as this engagement has on the **/engagement** call. The default value is *businessUnitID* from the /engagement request. This parameter is the same as the *agentGroupID* parameter and is used only to determine hours of operating in cases where both *agentGroupID* and *businessUnitID* are set next: get this data by *agentGroupID* value first and after by *businessUnitID* parameter. If both business unit level and agent group level settings are configured, then only agent group level settings are used. Hours of operation value is always true if this field is absent and *agentGroupID* is also not set.	Integer	No
customVariables	Can be used to store information that can be used for targeting	N/A	*No*

Name	Value	Type	Required
customVariables:action	Action to be performed on this custom variable	Must be one of ***append***, ***set***, ***remove***. Default is ***append.***	*No*
customVariables:name	Name of this custom variable	String	*Yes*
customVariables:persistence	Recommended persistence lifespan	Must be either ***volatile*** or ***persistent***. Default is ***persistent.***	*No*
customVariables:value	Value of this custom variable. Can be multi-valued	String	*Yes*
engagementID	Id of engagement provided by *Nuance* platform. This parameter is used to log into ***customAttribute*** record with attribute type ***engagementAttribute***.	Integer	*Yes*
sessionID	Id of current session. Must be the same as this engagement had on ***/engagement*** call. Required if the request contains applicationAttributes. This parameter is used to log into ***customAttribute*** record with attribute type ***application***.	String, possible values are: [a-z], [A-Z], [0-9], [-], [_]	*No*
siteID	Id of site provided by *Nuance* and will be a static value for client application. Must be the same as this engagement had in ***/engagement*** call	Integer	*Yes*
tcCustomerID	*customerID* provided by the *Nuance Engagement API* when the ***/engagement*** endpoint is called or by the *Nuance Data Collection API* when the ***/startSession*** endpoint is called. Must be the same as this engagement had on ***/engagement*** call in *customerID* parameter. Also, this parameter is used to do correct balancing	String, possible values are: [a-z], [A-Z], [0-9], [-], [_]	*Yes*
viewID	Identifier for the current content displayed (or otherwise provided) to the customer	String	*No*

Request Example 1

POST https:// api.touchcommerce.com/engagementAPI/v2/customer/event/assisted
POST body:

```
{
  "tcCustomerID": "465657113802851330",
  "siteID": 306,
  "businessUnitID": 22,
  "agentGroupID": 10004091,
  "viewID": "6789231234",
  "sessionID": "55552456234234",
  "engagementID": 123433332111,
  "businessRuleID":"12351dsf123213",
  "customVariables": [
    {
      "name": "attributeName@&^",
      "value": "attributeValue",
      "persistence": "persistent",
      "action": "append"
    }
  ],
  "attributes": [
    {
      "name": "visAttr 1",
      "value": "visAttr Value11",
      "attributeType": "visitorAttribute",
      "externalCustomerID": "Yes",
      "action": "set"
    },
    {
      "name": "attr1",
      "value": "visAttrValue22 !$>",
      "attributeType": "applicationAttribute",
      "action": "set"
    },
    {
      "name": "attr2",
      "value": "visAttrValue22 !$>",
      "attributeType": "engagementAttribute",
      "action": "remove"
    }
```

```
  ]
}
```

Response Example 1

```
200 OK

ETL logs example:

1495206882128-
customer.customAttribute?attributeType=visitor&customerID=465657113802851330&siteID=306&attr=visAttr+1,vis
Attr+Value11,set&pageName=6789231234&externalCustomerIDs=visAttr+1
1495206882128-
customer.customAttribute?attributeType=application&customerID=465657113802851330&siteID=306&sessionID=5555
2456234234&attr=attr1,visAttrValue22+%21%24%3E,set&pageName=6789231234
1495206882129-
customer.customAttribute?chatID=123433332111&attributeType=engagementAttribute&customerID=4656571138028513
30&siteID=306&attr=attr2,visAttrValue22+%21%24%3E,remove&pageName=6789231234
1495206882129-
customer.customAttribute?attributeType=variables&customerID=465657113802851330&siteID=306&vars=attributeNa
me%40%26%5E,attributeValue,append,persistent&pageName=6789231234
1495206882129-
conversionFunnel.assisted?incAssignmentID=465657113802851330&businessRuleID=12351dsf123213&chatID=12343333
2111&customerID=465657113802851330&siteID=306&sessionID=55552456234234&type=APIchat&pageName=6789231234&in
HOP=1
```

Request Example 2

POST https:// api.touchcommerce.com/engagementAPI/v2/customer/event/assisted

POST body:

```
{
  "tcCustomerID": "465657113802851330",
  "siteID": 306,
  "engagementID": 123433332111,
  "businessRuleID":"asdfasdasd4111"
}
```

Response Example 2

200 OK

ETL log example

```
1495206995659-
conversionFunnel.assisted?incAssignmentID=465657113802851330&businessRuleID=asdfasdasd4111&chatID=12343333
2111&customerID=465657113802851330&siteID=306&sessionID=465657113802851330&type=APIchat&inHOP=1
```

Conversation Transcript (GET)

Get Conversation is used to get the transcript of the last or all conversations to show the customer by customerID or persistentCustomerID. The persistentCustomerID parameter is in JSON format and includes private customer data such as an email address, phone number, or account ID.

> ***For Example:*** If filter is "${phone}" IN persistentCustomerID.phone then if we do not pass persistentCustomerID and it will return nothing.
>
> If the filter is "${email}" IN persistentCustomerID.email OR customerID = "${customerID}" and if we do not pass pcID then it will return data only by customerID.
>
> If there is no record for site in conversation_filter_settings then it will use customerID for filtering.
>
> Note. Please get in touch with the Nuance Dev team to set up conversation_filter_settings.
>
> It is only for asynchronous chats users. Cannot be used for non-async chats to get transcripts.

This returns a list of messages for the last conversation or messages for each conversationID (all mode.) It also returns initialCustomerID (customerID started conversation) in response cross-device or cross-browser support.

URI	conversationTranscript
HTTP Method	GET
Formats	XML or JSON

Parameters

Name	Value	Type	Required
authenticatedUser	JSON object of persistent customer IDs. 1. The authenticatedUser object will consist of persistentCustomerID object which will itself be an object. 2. The authenticatedUser object can have additional fields that are client-specific that can be used to verify the user.	string	no
businessUnitID	Initial Business Unit of current conversation	string	no

Name	Value	Type	Required
customerID	CustomerID value returned in engagement request API call.	string	yes
conversationObjectID	The objectID returned by the token-service after token verification call. Used to fetch conversation data. Must be passed in a pair with conversationTokenID.	string	no
conversationTokenID	The tokenID generated by the token-service for the a third-party channel conversation and passed in the landing page URL to continue a conversation. Must be used in a pair with conversationObjectID. If mode is LAST, then conversation attached to the token will be fetched for the verified token. If the token is not valid, then the last customerconversation will be fetched. If mode is ALL, then all customer conversations along with conversation linked to the token will be returned.	string	no
days	Number of days for which conversation transcript is requested. It is used only with mode=ALL.	string	no
mode	Accepts values: all and last. Last is default	string	no
output	XML or JSON. Default is XML.	string	no

Name	Value	Type	Required
persistentCustomerID	Private customer identifier from client side in JSON format Example: {"email":"a@a.com","phone":"123"} **NOTE**: if any value in JSON key-value pair is empty then request will fail. ***Here is an Example*** of an *invalid* persistentCustomerID. {"email":"", "phone":"111-11-11"} If a conversation filter setting is not specified, then filtering by customerID is used. **IMPORTANT**: When client filters by persistent customer ID it is important to set **unique** persistentCustomerID per real client (usually when customer logs in set any specific data like ID). Potential consequence of using the same persistent customer ID for different customer is possibility to see historic transcript of another customer with the same persistent customer ID even on another device and another customerID. ***Here is an Example*** of a bad persistentCustomerID: persistentCustomerID={“name”:”John”} In this case every customer with the name John will be able to see transcripts of each other. Best practice is to use a unique ID of customer based on login data: persistentCustomerID={"id":"123484398578395389457938475983473949590"}	string	no
siteID	Site ID to check that Site ID is corresponded for user	string	yes

Note. If the client uses OAuth authorization, then valid access token should be sent in the “Authorization” header. Otherwise the client should log into the API as engagement API user.

Example of sending a message:

GET https://api.touchcommerce.com/engagementAPI/v2/customer/conversationTranscript?siteID=306&customerID=1111111&businessUnitID=22&output=json&persistentCustomerID={%27email%27:%20%27aa@aa.aa%27}

Response Example

```
200 OK

Body: "messages":[{"messageText":"I need some
help","messageType":"chatLine"},{"messageText":"hi","agentID":"agt@tc.com","messageType":"chatLine","agent
.alias":"agt@tc.com"},{"messageText":"bye","agentID":"agt@tc.com","messageType":"chatLine","agent.alias":"
agt@tc.com"},{"messageText":"I need some
help","messageType":"chatLine"},{"messageText":"hello!","agentID":"agt@tc.com","messageType":"chatLine","a
gent.alias":"agt@tc.com"},{"messageText":"bye!","agentID":"agt@tc.com","messageType":"chatLine","agent.ali
as":"agt@tc.com"}]}
```

Customer Start and Stop Typing

Activity types customerStartTyping and customerStopTyping tell the agent when the customer starts or stops typing. Start typing should only be repeated if there was a customerStopTyping request. The customerStopTyping request should be sent when there is no user input for the timeout period (e.g. 4 seconds).

URI	customerStartTyping and customerStopTyping
HTTP Method	POST
Formats	XML or JSON

Parameters

Name	Value	Type	Required
activityType	Activity types include the following: customerStartTyping, customerStopTyping, customerMinimizeWindow, customerRestoreWindow and uploadFile.	string	Yes
customerID	The customerID value returned by the Engagement Request API call. This parameter is required for correct load balancing.	string	Yes
engagementID	The engagementID value returned by the Engagement Request API call.	string	Yes
messageText	Custom text for activity used only for uploadFile activity.	string	Yes for uploadFile

Message Example

```
POST https://api.touchcommerce.com/engagementAPI/v2/customer/activity?customerID=111&engagementID=111

body:activityType=customerStartTyping
```

Response Example

```
200 OK
```

Pseudo Code

The following pseudo code sketches the algorithm for the client to send start and stop typing:

```
boolean isTyping = false;

Timer stopedTypingTimer = null;

// event handler for user typing event (onKeyPress, onChange or similar event)
function onUserTyping() {
  // reschedule the stoppedTyping timer to the new time = now + 4seconds
  cancelTimerIfNotNull(stoppedTypingTimer);
  stoppedTypingTimer = scheduleTimer(now + 4sec, onStoppedTyping);

  if (!isTyping) {
    POST activity?activityType=customerStartTyping
    isTyping = true;
  }
}

function onStoppedTyping() {
  isTyping = false;
  POST activity?activityType=customerStopTyping
}
```

Customer Active Conversations (GET)

Used to check if there are active conversations and return conversationIDs and additional data without a transcript.

This is only for asynchronous chat users.

URI	customerActiveConversations
HTTP Method	GET, POST
Formats	JSON

Parameters

Name	Value	Type	Required
businessUnitID	Business unit ID of the conversation. If this endpoint is not passed then settings will be requested by siteID.	string	yes
customerID	Customer ID value returned in engagement request API call. If conversation filter setting is present, then patter from setting will be used. Otherwise data will be filtered by customerID. If it is not passed in this case, then no data will be returned.	string	no

Name	Value	Type	Required
persistentCustomerID	Private customer identifier from client side in JSON format **For Example:** {"email":"10112018_opera@tc.com"} **NOTE**: if any value in the JSON key-value pair is empty then request will be failed. Following example of invalid persistentCustomerID: {"email":"", "phone":"111-11-11"} Used for filtering. Passed to template from conversation filter setting. If conversation filter setting is not specified then customerID is used **IMPORTANT**: When client use conversation filter by persistent customer ID it is important to set **unique** persistentCustomerID per real client (usually when customer logs in set any specific data like ID). Potential consequence of using the same persistent customer ID for different customer is possibility to see data of another customer with the same persistent customer ID even on another device and another customerID. **For Example:** Bad persistentCustomerID : persistentCustomerID={"name":"John"} In this case every customer with the name John will be able to see data other. Best practice is to use the unique ID of the customer based on login data: `persistentCustomerID={"id":"12348439857839538945793847598347394 9590"}`		
siteID	Site ID to check that Site ID is corresponded for user	string	yes

Example

```
GET https:// api.touchcommerce.com/engagementAPI/v2/customer/customerActiveConversations?siteID=306&custom
erID=1234567&persistentCustomerID={%27email%27:%20%27aa@aa.aa%27}
```

Response Example

```
200 OK

Body:{hasPendingMessages”: true}
```

Customer Status (GET)

Used to check pending messages for the current customer. Filtering is by persistent CustomerID. This endpoint checks whether agent offline pending messages exist for customer. This is only for asynchronous chat users.

URI	customerStatus
HTTP Method	GET
Formats	XML or JSON

Parameters

Name	Value	Type	Required
customerID	The customer ID from the chat session	string	no
conversationTokenID	The tokenId generated by the token-service for third-party channel conversation and passed in landing page URL to continue the conversation. Must be passed in pair with conversationObjectID. All customer pending conversations messages data along with conversation linked to token will be returned.	string	no
conversationObjectID	The objectId returned by the token-service after token verification call. Used to fetch conversation data. Must be passed in pair with conversationTokenID.	string	no
output	XML or JSON. Default is XML.	string	no

Name	Value	Type	Required
persistentCustomerID	Private customer identifier from client side in JSON format Ex: {"email":"10112018_opera@tc.com"} . **NOTE**: if any value in JSON key-value pair is empty then request will be failed. Following example of invalid persistentCustomerID: Unknown macro: {"email"} If conversation filter setting is not specified, then customerID is used **For Example:** Bad persistentCustomerID : persistentCustomerID={"name":"John"} In this case every customer with the name John will be able to see data other. Best practice is to use the unique ID of the customer based on login data: persistentCustomerID={"id":"123484398578395389457938475983473949590"}	string	yes
siteID	Site ID to check that Site ID is corresponded for user		

Example of sending a message:

```
GET https:// api.touchcommerce.com/engagementAPI/v2/customer/customerStatus?sit
eID=306&output=json&persistentCustomerID={%27email%27:%20%27aa@aa.aa%27}
```

Response Example

```
200 OK
Body: {"hasPendingMessages": true}
```

Data Pass to Agent

Once an engagement has been established and an agent is assigned, you can use this API to send data to the agent for context about the user or their experience. Data pass can occur at any time during an engagement and is displayed to the agent or agents within the Agent Interface transcript panel.

Examples of information that might be sent to the agent are emailID, a client provided customer identifier, customer demographics, etc.

Sensitive data values can be marked for masking as well. Data marked for masking will be displayed clearly to the agent but replaced with X's before being saved to transcripts.

URI	dataPass
HTTP Method	POST
Formats	XML or JSON

Parameters

Name	Value	Type	Required
agentID	Agent which will see dataPass	string	yes
<attributeName1>	<attributeValue1> Any POST parameter except reserved names At least one attribute parameter is required.	string	yes
<attributeName2>	<attributeValue2>	string	no
engagementID	ID passed for this engagement by the Engagement API response.	string	yes

Masking

Any attribute values can be marked for masking by the following syntax.

```
((masked{['myDataString']}masked))
```

Request Example 1

```
POST https:// api.touchcommerce.com/engagementAPI/v2/customer/dataPass

POST Body:

engagementID=86448213037320856&agentID=2123411123&key1=value1&key2=value2&key3=value3
```

Request Example 2

```
POST https://api.touchcommerce.com/engagementAPI/v2/customer/dataPass

POST Body:

engagementID=86448213037320856&agentID=2123411123&key1=((masked{['value1']}masked))&key2=value2&key3=value
3
```

Response Example

```
200 OK
```

Email Transcript

The Email Transcript call allows a chat customer to receive a transcript of their chat content via email. When emailTranscript is called, the client may flag an open engagement to be sent to the specified customer email address once the engagement is complete. The engagement is considered completed when the session has timed out or the client uses the API to end the chat using stateChange=closed. When the engagement is complete, the email containing the transcript will be queued up to be sent via email.

URI	emailTranscript
HTTP Method	POST
Formats	XML or JSON

Parameters

Name	Value	Type	Required
customerID	The customerID value returned by the Engagement Request response.	string	yes
customerName	Default = **You** if a customer name is not specified.	string	no
engagementID	The engagementID value returned by the Engagement Request response.	string	yes
emailAddress	Must match a valid email address.	string	yes
emailSpecID	The email specification ID that describes the email criteria defined in TouchPortal. In case of a missing or invalid emailSpecID, the default site emailSpec will be used.	string	no
output	The accept header value is ignored. The default value is xml.	string	no
siteID	Matches the siteID. Missing or invalid siteID will generate a 403 error.	string	yes

Request Headers

Name	Value	Type	Required
X-Protection-Id	Custom protectionID value will be used if protection mode is enabled. Parameter is now allowed when protection mode is not enabled.	string	no

Request Example

```
POST https:// api.touchcommerce.com/engagementAPI/v2/customer/emailTranscript

POST body:

customerID=12334112&engagementID=39747223223&emailAddress=janesmith@gmail.com&siteID=402
```

Response Example

```
200 OK
```

Engagement Request

The purpose of the Engagement Request call is to request an engagement (e.g. “chat session”) be initiated. Whether or not an Engagement Request will be initiated is also dependent on Agent availability. Agent availability is based on the following algorithm:

$$(Queue\ Threshold) * (Total\ Agent\ Slots) - ((Total\ Active\ Engagements) + (Total\ Queued\ Engagements))$$

The primary return value is status, which indicates whether the request was immediately assigned to an agent (accepted), placed in queue (queued) or refused (denied).

Ideally, an Engagement Request call should not be made until there is truly a need to assign/reserve an agent slot for a customer. For example, it should not be called before presenting a click to chat or proactive invitation. Rather it should be called when the customer engages chat by sending a first message.

URI	engagement
HTTP Method	GET or POST
Formats	XML or JSON

Parameters

Name	Value	Type	Required
agentAttributes	List of agent attributes used in the routing process. Format: attribute1, value1;attribute2,value2;attribute3,value3 Example: location,SD;location,AH;timezone,pst **Note:** Agent attributes are used to specify the attributes used for routing (sometimes called routing attributes). If set, this parameter will include the named attributes when checking availability or requesting an engagement. This call will look for an agent who possesses all of the listed skills.	string	No

Name	Value	Type	Required
agentGroupID	Agent Group ID to check agent availability. This parameter is required only for sites that have agent groups.	string	Yes for sites where Agent Group functionality is enabled. No for sites where Agent Group functionality is not enabled.
agentMessageEndpointID	Engagement's agent message to be posted to the customer specified endpoint instead of long polling. Based on **agentMessageEndpointId** provided, specific endpoint URL is fetched from DB along with authentication parameters. Endpoint data for the site can be configured through Portal UI.	integer	no
agentMessageEndpointDynamicUrl	Dynamically provided Callback URL to POST agent messages to client. If dynamic URL is provided, instead of polling for agent messages CEAPI will post the agent messages to callback URL whenever the agent replies. In order to enable this feature callback URL should satisfy at least one Regexp. Regexp can be configured on portal for a particular Site. ***Note:*** 1) Only supported for CEAPI V2 version. 2) Combination of **agentMessageEndpointId** and **agentMessageEndpointDynamicUrl** can be used for allowing dynamic URL functionality as well as authentication feature.	string	no
asyncSurveySpecID	ID of specification to be used to schedule post chat survey for asynchronous chats when conversation is resolved or expired.	string	no

Name	Value	Type	Required
authenticatedUser	JSON object of persistent customer IDs. 1. The authenticatedUser object will consist of persistentCustomerID object which will itself be an object. 2. The authenticatedUser object can have additional fields that are client-specific that can be used to verify the user.	string	no
automatonDataMap	Automaton parameters that will be passed to AI automaton and integration automaton. Format of this field is simple JSON object string. **For Example:** {"tGuardToken":"ePzER6z76Stll7HV1gl7", "field2":"value2", ...}	string	no
automatonFields	Extension for automaton content which is used for automaton initialization.	string	no
automatonID	Associated automation ID with this engagement request. Currently used for virtual agent integration. Virtual agent automaton ID must be pre-defined in the configuration for this parameter.	number	no
browserType	Customer's browser type.	string	no
browserVersion	Customer's browser version.	string	no
businessRuleAttributes	List of business rule attributes used for ETL logging. Format: attribute1, value1; attribute2, value2. (Used to log Facebook or Twitter account name, message type and start keyword (if it is not empty). Facebook message types: post, comment, reply. Twitter message type: message.	number	no
businessRuleID	Name of the business rule. This parameter is used to tie this engagement to a business rule for reporting purposes.	string	no
businessUnitID	Business Unit ID is used to tie this engagement to a business unit for reporting purposes	string	yes

<table>
<tr><th>Name</th><th>Value</th><th>Type</th><th>Required</th></tr>
<tr><td>customerID</td><td>If the application has a customerID from a prior request to this API, then it should be passed here. If this field is left blank, a new customerID will be created and returned.</td><td>string</td><td>no</td></tr>
<tr><td>conversationChannel</td><td>
<table>
<tr><th>Possible Values</th><th>Access</th><th>Description</th></tr>
<tr><td>SMS_TWILIO</td><td rowspan="6">Private</td><td rowspan="6">The conversationID parameter is required.</td></tr>
<tr><td>FACEBOOK</td></tr>
<tr><td>TWITTER</td></tr>
<tr><td>APPLE_BUSINESS_CHAT</td></tr>
<tr><td>NINA_WEB_API_PROXY</td></tr>
<tr><td>SMS_NUANCE</td></tr>
<tr><td>CEAPI</td><td rowspan="2">Public</td><td>Default value</td></tr>
<tr><td>WEB</td><td></td></tr>
</table>
</td><td>string</td><td>no</td></tr>
<tr><td>conversationId</td><td>This parameter is required when conversationChannel values are equal to SMS_TWILIO, FACEBOOK, TWITTER, APPLE_BUSINESS_CHAT, NINA-WEB_API_PROOXY, SMS_Nuance.

This is a private parameter. Possible values are [a-z], [A-Z], [0-9], [-], [_].</td><td>string</td><td>no</td></tr>
<tr><td>createNewConversation</td><td>If the value is true, then always create a new conversation for any new engagement. This parameter is only for asynchronous. Engagement (isAsyncEngagement=true).</td><td>boolean</td><td>no</td></tr>
<tr><td>customerName</td><td>Custom name of the customer as it displays in the Agent Interface</td><td>String</td><td>no</td></tr>
<tr><td>deviceType</td><td>Device type which the customer uses to start this engagement. Default value is CEAPI on the reporting side and EAPI in the Agent Interface side.</td><td>string</td><td>no</td></tr>
</table>

Name	Value	Type	Required
fakeEngage	This flag is set to true in to create a chatroom without routing to an agent. Used to add a customer message to a conversation transcript.	boolean	no
followUpAgentId	Follow Up agent ID is used to start a follow up chat for the follow up mode TIMEOUT.	string	no
followUpMode	Follow Up Mode indicates how the follow up chat was started. followUpMode – etlValue • Agent – 1 • Timeout – 2 • Client – 3	string	no
initialCustomerID	Customer ID used to continue a conversation when the client changes conversation channel and the current customer ID (customerID parameter) is different from the initial conversation.	string	no
InitialMessage	Initial message sent by customer.	string	yes
initialMessageReply	Reply to client with initial message (text from InitialMessage parameter) in one of the GET /message responses. Default is true.	boolean	no
intactAutomatonAnswer	If the value is true, the message from the virtual assistant is NOT wrapped in the div automaton. If “false”, "<div onclick=\"" + CI_ON_CLICK_HANDLER + "\" >" + agentAnswer + " ".	boolean	no
isAsyncEngagement	If the value is true, the engagement is asynchronous.	boolean	no
isTopicsEnabled	This parameter is only used with asynchronous engagements (isAsyncEngagement=true). See isAsyncEngagement parameter. The main idea of topic functionality is to support customers being able to have several active conversations at a time and be able to restart a conversation from the past. If the value is true, the engagement supports topic functionality. If the value is false (by default), all engagements for the customer will be in the scope of one default topic according to channel type.	boolean	no
launchPageMarker	Chat title name that the agent will see in the Agent Interface on the chat tab. The default value is EAPI.	string	no

Name	Value	Type	Required
launchType	launchType displays in the Agent Interface to notify the Agent which type of device or experience the customer is in. To set Agent Interface background color, use these launchType values in all CAPS: • For color green; launchType = SMS (for SMS) • For color yellow; launchType = DTT (for tablet) • For color pink; launchType = DTSP (for mobile)	string	no
opener	Opener message that will be added to transcript upon agent assign.	string	no
operatingSystemType	Customer's operating system.	string	no
output	XML or JSON. Default is XML.	string	no
pageID	Number that identifies the web page where the engagement request starts. This parameter must be pre-defined in Nuance Portal.	number	no
persistentCustomerID	Persistent (unique) customer ID.	string	no
phoneNumber	If this engagement is to be a call engagement, use this field to pass the customer's phone number to the agent.	number	no
priority	Initial engagement priority of the engagement request. A higher number indicates a higher priority in the queue.	number	no
protectionModeEnabled	If value is true, it is possible to send own protectionId in the header or to get protectionId generated via the system.	boolean	no
pushNotificationDeviceID	Device ID for push notification functionality. The previous deviceID value should be retained in cases where the devieID is not provided in the engagement request call.	string	no
queueMessagingSpecID	The ID of the queue messaging spec as identified in TouchPortal. If the queue messaging spec is valid, it will pull the proper queue message and display it as message text when the API caller calls the Get message API call.	number	no
queueThreshold	Queue threshold value for engagement. Used to calculate whether this engagement goes into the queue in cases where agent slots are not available.	number	no
scriptId	The script tree ID.	string	no
sessionID	ID of the current session. Default value is tcCustomerID value from this request. Possible values are: [a-z], [A-Z], [0-9], [-], [_]	string	no
siteID	Site ID to check agent availability	string	yes

Name	Value	Type	Required
specificAgentID	ID of the preferred agent to start the conversation with. **Note.** If you use this parameter, in most requests the response will be 200 (accepted. But you will get the real status of the chat in GET /message. It can be2nd to 5th message. If you get chat.accepted, just continue the chat. If you get chat.denied, you should send exitChat request.	string	no
topicID	Topic ID in which the customer wants to continue engagement in case of isTopicsEnabled=true. If the parameter is omitted in case of isTopicsEnabled=true, then new topicID is generated. In cases where isTopicsEnabled=false, all engagements will be in the scope of one default topic according to channel type. For example: channel type - topicID: • SMS_TWILIO-1 • FACEBOOK-2 • TWITTER-3 • APPLE_BUSINESS_CHAT-4 • CEAPI-5 • NINA_WEB_API_PROXY-6 • WEB-7 • SMS_NUANCE-8	string	no

Name	Value	Type	Required
topicName	Name of the topic of case isTopicsEnabled=true. In cases where isTopicsEnabled=false, all engagements will be in scope of one default topic according to channel type. The name of topic is according to channel type. For example: channel type - topicName: • SMS_TWILIO-SMS_TWILIO • FACEBOOK-FACEBOOK • TWITTER-FACEBOOK • APPLE_BUSINESS_CHAT-APPLE_BUSINESS_CHAT • CEAPI-CEAPI • NINA_WEB_API_PROXY-NINA_WEB_API_PROXY • WEB-WEB • SMS_NUANCE-SMS_NUANCE	string	no
virtualAgent.NinaVars	Allows the customer to pass data to the virtual assistant. NinaVars (datapass to Nina) variables should be sent in a JSON string in the URL parameter. This will override automatonFields property i.e. if virtualAgent.ninaVars is present, only that is send to VA.	string	no
virtualAgent.*	The '*' (asterisk) is a keys sequence that represents the path in TalkAgentRequest where the specific data should be inserted. Both value and TalkAgentRequest are JSON objects. The last item in the keys sequence should be the same as the first key in value. The value should be URL encoded.	string	no
visitorAttributes	List of visitor attributes used for reporting. Format: attribute1, value 1;attribute 2, value 2; attribute3, value 3 ***Example:*** location,SD;location,AH;timezone,pst	string	no

Request Headers

Name	Value	Type	Required
X-Protection-Id	Custom ProtectionId value which will be used if protection mode is enabled. This parameter is not allowed when protection mode is not enabled.	string	no

Response Values

Name	Value	Type	Description
customerID	New customerID is provided if a customerID is not passed with this request	string	A customer identifier provided by RightTouch. This ID is the first time an engagement request is made for a unique customer. This value should be persisted in the client application and included with all subsequent requests.
engagementID	The Id given to this specific engagement.	string	A unique identifier for a chat session. It is set and returned with this response. Must be sent by client with each subsequent request during a chat session.
protectionID	protectionID generated via the system or from header request.	string	A unique identifier for additional protection of the chat. It can be generated via system or customer can send their own protectionID via engagement request header.

Name	Value	Type	Description
status	accepted | queued | denied	string	Indicates whether the request is assigned, placed in queue or refused. The “denied” value could be returned in case of no availability or out of HOP or wrong parameters. For example, a non-existing queueMessagingSpecID. For Async chats, the “denied” status could be returned in case of wrong parameters, in case of no agent availability or out of HOP it will ***not*** return a “denied” value. For example, a non-existing queueMessagingSpecId or a wrong configuration of the conversation settings in the database. Note. For async chats, it returns status only according to agent availability for passed queue key (agentGroupID + businesUnitID) but it does not calculate preferredAgent or possibly existing chatroom in conversation so this status could be false.

Example Virtual Agent Datapass / Automaton Fields

```
GET https://api.touchcommerce.com/engagementAPI/v2/customer/engagement?siteID=306&businessUnitID=22&Initial
&virtualAgent.NinaVars={"key1":"ABC","key2":"123"}
```

```
https://api.touchcommerce.com/engagementAPI/v2/customer/engagement?siteID=10000000&businessUnitID=19001192
&automatonFields={%22datapass%22:{ "key1":"ABC","key2":"123"}}
```

Request Example 1

```
GET https://api.touchcommerce.com/engagementAPI/v2/customer/engagement?siteID=306&businessUnitID=22&Initia
lMessage=I%20need%20some%20help
```

Response Example 1

```
200 OK
```

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?><message><status>accepted</status><customerID>7657620078025131029</customerID><engagemen
tID>7657620078025405816</engagementID></message>
```

Request Example 2

POST https://api.touchcommerce.com/engagementAPI/v2/customer/engagement

POST body:

```
siteID=306&businessUnitID=22&InitialMessage=I%20need%20some%20help
```

Response Example 2

```
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><message><status>accepted</status>
<customerID>7657620078025131029</customerID><engagementID>765762007802540581 6</engagementID></message>
```

Response Example 2

```
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><message><status>accepted</status>
<customerID>7657620078025131029</customerID><engagementID>765762007802540581 6</engagementID></message>
```

Exit Chat

Exit chat is a POST call that shows an exit from an engagement that is closed by the customer. This action closes the chat for all actions from the customer's side

URI	exitChat
HTTP Method	POST
Formats	XML or JSON

Parameters

Name	Value	Type	Required
customerID	The customerID value returned by the Engagement Request response.	string	yes
engagementID	The engagementID value returned by the Engagement Request response.	string	yes

Request Example

```
URL https://api.touchcommerce.com/engagementAPI/v2/customer/exitChat

POST body

customerID=566534534634534&engagementID=52362345623453
```

Response Example

```
200 OK
```

Message (GET)

GET Message is used to monitor the status of the engagement, collect agent messages, etc. The client application should endeavor to have a GET request pending at all times. A GET message should be sent immediately following a successful Engagement Request and each GET request response for which state is not closed. This request should be sent one after another to avoid losing the engagement. Otherwise the server connection will close the engagement due to the 60 second timeout.

GET Message also supports Callback functionality where engagement details are pushed to a customer endpoint. The Callback response values are a subset of the Message GET response value superset. Callback parameters are indicated in the Response Values table.

Descriptive examples of information returned by GET Message are listed here:

- Nothing pending (In this case the server will delay the response for several seconds waiting to see if a status change occurs). If a status change occurs, a response is sent immediately with the new status (e.g. a new message from the agent). If no status change has occurred for 30sec, a 204 HTTP status code is returned.
- Queue status – Callback functionality supported
- A new agent message – Callback functionality supported
- Agent is typing – Callback functionality supported

URI	message
HTTP Method	GET
Formats	XML or JSON

Parameters

Name	Value	Type	Required
customerID	The customerID value returned by the Engagement Request API call. Parameter is required for correct balancing.	string	yes
engagementID	The engagementID value returned by the Engagement Request API call.	string	yes

Name	Value	Type	Required
instantResponse	true or false. If true, then return response immediately without waiting	boolean	no
output	XML or JSON. Default is XML.	string	no
requestEntireTranscript	True or false. If true, the entire transcript for the specified engagement will be returned in a single response. If false, one message from transcripts will be returned.	boolean	no

Request Headers

Name	Value	Type	Required
X-Protection-Id	The ProtectionId value returned by the Engagement Request response.	string	no

Response Values

Name	Value	Type	Callback Functionality	Description
agent.alias	Alias of the agent.	string	no	Alias of the agent.
agentName	Name of agent assigned to this engagement	string	no	Name of the agent.
agentID	Unique ID of the agent assigned	string	Yes	The unique identifier of the agent. (Same as agent login ID)
agentIsTypingState	true / false **Note:** agentIsTypingState is ONLY applicable for PUSH notification.	boolean	Yes	Indicates when the agent has started or has stopped typing.

Name	Value	Type	Callback Functionality	Description
agentAttributes	URL encoded agent attributes	string	no	The value may be available only when state = assigned. The value can be used as the availableAgentAttributes parameter to /showSurvey for logging automaton.started keySurvey post chat survey.
command	pushURL	string	Yes	A pushURL command is sent (used for sending links).
customerID	Value returned in engagement request API call. This parameter is required for correct load balancing.	string	no	Unique customer ID.
engagementID	Unique numeric ID of current chat	string	Yes	The unique identifier of the current chat.
escalate	true / false	boolean	no	Indicates that a chat has been escalated.
from	Message sender	string	Yes	This field contains the recommended customer facing agent name (typically presented something like “Jim: How can I help you?” where Jim is the agent name presented to the customer.
message	Tag to wrap the content of each individual message		no	Indicates a new agent message is included
messageData	This field represented in string format is used to pass rich content to the customer such as jsonString, jsonObject, HTML tags, etc.	string	no	Rich content data to be rendered to the customer. For simple strings, we use ‘messageText’ field but for anything more complex than that such as HTML tags, jsonString, jsonObject etc. we use messageData to pass to the customer.

Name	Value	Type	Callback Functionality	Description
messageText	Chat line text	string	Yes	The agent text/message.
messageType	chatline | command | stateChange | chatroom.member_connected | chatroom.member_lost	string	Yes	Type of message that is being returned. When the “stateChange” message type is received, you will also receive the “state” field indicating what state was changed. When the “command” message type is received, you will also receive the “command” field indicating what command was sent.
ninaResponse.extended :alternateText:text	Plain text	string	no	Extended Nina response (optional) which could be shown to end user as alternative to default Nina response.
ninaResponse.extended alternateText2:text	Plain text	string	no	Extended Nina response (optional) which could be shown to end user as alternative to default Nina response. Suitable to be converted to audio and played on end user side.
queueDepth	Number of customers ahead of you in queue.	number	Yes	Queue depth is provided when state is queued.
sequenceNumber	Sequence number of the message.	number	Yes	The client application can use the sequence number to ensure that messages are displayed in the correct order.
state	assigned | queued | closed | agentIsTyping| transfer | conference	string	Yes	The current status of the engagement. This field presents only if **messageType = stateChange**.

Name	Value	Type	Callback Functionality	Description
agentIsTypingState	true/false **Note:** agentIsTypingState is ONLY for PUSH notification.	boolean	Yes	Indicates when the agent has started or has stopped typing.
url	URL of page sent	string	Yes	When messageType is the command, this field will be included, and it will contain the URL to be pushed into the customer's browser.
waitTime	Estimated wait time in seconds	number	Yes	Estimated wait time is provided when state is queued.

Internal Properties

The system returns these properties but please do not use them. All of these properties are internal and can be deleted or renamed at any time.

Name	Value	Type	Description
agent.alias	Alias of the agent.	string	Alias of the agent.
business_unit.id	Numeric ID assigned to chat business unit.	string	Numeric ID assigned to chat business unit.
chat.receiver_id	Numeric ID of the message receiver.	string	The unique ID of the message receiver.
chatroom.chat_id	Unique numeric ID of the current chat	string	The unique identifier of the current chat.

Name	Value	Type	Description
chatroom.member.id	Numeric ID of member from state change action.	string	Numeric ID of member from state change action.
chatroom.member.name	Name of member from state change action.	string	Name of member from state change action.
chatroom.member.type	Type of member who sent this message.	string	Type of member who sent this message.
client.display.text	Text with chat details about state change action for client.	string	Text with chat details about state change action for client.
cobrowse.enabled	true / false	boolean	Shows whether cobrowse functionality is enabled for current chat.
display.text	Text with chat details about state change action.	string	Text with chat details about state change action.
reason	Reason of state change message	string	Reason of state change message
screening	true / false	boolean	Shows conference mode: screening or full
tc.mode	transfer | conference		Shows transfer mode: transfer, conference
user.id	Numeric ID of the message sender.	string	Unique ID of the message sender.

Internal parameter returns from CR

Name	Example	Message types where this parameter appears
agent.alias	<agent.alias>Agent</agent.alias>	<messageType>stateChange</messageType><state>assigned</state>, <messageType>chatLine</messageType>, <messageType>chatroom.member_connected</messageType>
business_unit.id	<business_unit.id>22</business_unit.id>	<messageType>stateChange</messageType><state>assigned</state>, <messageType>chatroom.member_connected</messageType>
chatroom.chat_id	<chatroom.chat_id>7657620082880181644</chatroom.chat_id>	<messageType>chatroom.member_connected</messageType>, <messageType>chatroom.member lost</messageType>
chatroom.member.name	<chatroom.member.name>1</chatroom.member.name>	<messageType>chatroom.member_connected</messageType>
chatroom.member.id	<chatroom.member.id>1</chatroom.member.id>	<messageType>chatroom.member_connected</messageType>, <messageType>chatroom.member lost</messageType>
chatroom.member.type	<chatroom.member.type>agent</chatroom.member.type>	<messageType>chatroom.member_connected</messageType>, <messageType>chatroom.member lost</messageType>
chatrouter.address	<chatrouter.address>192.168.0.20</chatrouter.address>	<messageType>stateChange</messageType><state>assigned</state>
client.display.text	<client.display.text>Agent has left the chat&nl;You are being transferred, please hold...</client.display.text>	<messageType>stateChange</messageType><state>transfer</state>, <messageType>chatroom.member_connected</messageType>
cobrowse.enabled	<cobrowse.enabled>true</cobrowse.enabled>	<messageType>stateChange</messageType><state>assigned</state>

Name	Example	Message types where this parameter appears
display.text	<display.text>Agent is typing...</display.text>	<messageType>stateChange</messageType><state>agentIsTyping</state>, <messageType>chatroom.member_connected</messageType>, <messageType>chatroom.member_lost</messageType>, <messageType>stateChange</messageType><state>closed</state>
escalate	<escalate>false</escalate>	<messageType>stateChange</messageType><state>closed</state>
event.agent_first_name	<event.agent_first_name>User First Name</event.agent_first_name>	<messageType>stateChange</messageType><state>assigned</state>, <messageType>chatroom.member_connected</messageType>
event.agent_last_name	<event.agent_last_name>User Last Name</event.agent_last_name>	<messageType>stateChange</messageType><state>assigned</state>, <messageType>chatroom.member_connected</messageType>
exit	<exit>true</exit>	<messageType>chatroom.member lost</messageType>
host.node.id	<host.node.id>192.168.0.20</host.node.id>	<messageType>stateChange</messageType><state>assigned</state>, <messageType>stateChange</messageType><state>transfer</state>
msg.originalrequest.id	<msg.originalrequest.id>5722840798275623</msg.originalrequest.id>	<messageType>stateChange</messageType><state>assigned</state>, <messageType>stateChange</messageType><state>transfer</state>
owner	<owner>true</owner>	<messageType>chatroom.member_connected</messageType>
reason	<reason>Transfer accepted and sent to 1</reason>	<messageType>stateChange</messageType><state>transfer</state>
screening	<screening>true</screening>	<messageType>chatroom.member_connected</messageType>
script.id	<script.id>200023</script.id>	<messageType>chatLine</messageType>

Name	Example	Message types where this parameter appears
tc.mode	<tc.mode>transfer</tc.mode>	<messageType>chatroom.member_connected</messageType>, <messageType>chatroom.member_lost</messageType>, <messageType>stateChange</messageType><state>closed</state>
type	<type>2</type>	<messageType>stateChange</messageType><state>agentIsTyping</state>
user.id	<user.id>2</user.id>	<messageType>stateChange</messageType><state>agentIsTyping</state>

Request Example

```
GET https://api.touchcommerce.com/engagementAPI/v2/customer/message?engagementID=1234567890&customerID=9876543210
```

Response Examples

Chat-line example:

```
200 OK

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<messages>
  <message>
    <agentID>1</agentID>
    <agent.alias>TouchCommerce</agent.alias>
    <messageText>Hello</messageText>
    <messageType>chatLine</messageType>
    <engagementID>11352335599561178</engagementID>
  </message>
</messages>
```

Command example:

```
200 OK
```

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<messages>
  <message>
    <messageType>command</messageType>
    <command>push-url</command>
    <url>http://www.google.com
    <url>
      <time>1234567890</time>
    </url></url>
  </message>
</messages>
```

State change example when engagement is assigned to agent:

```
200 OK

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<messages>
  <message>
    <state>assigned</state>
    <agentID>2</agentID>
    <agent.alias>Agent</agent.alias>
    <messageType>stateChange</messageType>
    <engagementID>7657620082316703106</engagementID>
    <host.node.id>192.168.0.20</host.node.id>
    <business_unit.id>22</business_unit.id>
    <cobrowse.enabled>true</cobrowse.enabled>
    <chatrouter.address>192.168.0.20</chatrouter.address>
    <event.agent_last_name>User Last Name</event.agent_last_name>
    <event.agent_first_name>User First Name</event.agent_first_name>
    <msg.originalrequest.id>5722840234794661</msg.originalrequest.id>
  </message>
</messages>
```

State change example when chat is closed by the agent:

```
200 OK

<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<messages>
  <message>
    <state>closed</state>
    <escalate>false</escalate>
    <messageType>stateChange</messageType>
    <display.text>Agent '2' exits chat</display.text>
    <engagementID>7657620082327188868</engagementID>
  </message>
</messages>
```

Send Message

Used to send a new message from the customer to the agent. This call is also used to end the engagement.

URI	message
HTTP Method	POST
Formats	XML or JSON

Request Content Type
`application/x-www-form-urlencoded`

Note. Values in all key-value pairs in body must be URL encoded, e.g. messageText=I%20need%20some%20help

Parameters

Name	Value	Type	Required
customerID	The customerID value returned by the Engagement Request API call. This parameter is required for correct load balancing.	string	yes
engagementID	The engagementID value returned by the Engagement Request API call.	string	yes
messageData	{"key":"value"} response for rich data to be passed back to agent. **Note:** The string should be encoded in the request e.g. %7B%22key%22%3A%22value%22%7D	string	no
messageType	Specifies the type of the message. Possible values: "stateChange", default "chatLine "	string	no
state	Possible values: "closed"	string	no
virtualAgent.*	Here, * is a keys sequence that represents the path in TalkAgentRequest where the specified data should be inserted. Both value and TalkAgentRequest are JSON objects. The last item in the keys sequence should be the same as the first key in value.	string	no

Request Headers

Name	Value	Type	Required
X-Protection-Id	The ProtectionId value returned by the Engagement Request response.	string	no

Example of sending a message:

```
POST https:// api.touchcommerce.com/engagementAPI/v2/customer/message

POST body:

customerID=9876543210&engagementID=123456789&messageText=I%20need%20some%20help
```

Response Example

```
200 OK
```

Example of closing an engagement
Closing an engagement is done by calling the

Send Message endpoint with messageType=”stateChatne” and state =”closed”.

```
POST https:// api.touchcommerce.com/engagementAPI/v2/customer/message

POST body:

customerID=9876543210&engagementID=123456789&messageType=stateChange&state=closed
```

Response Example

```
200 OK
```

SetAuthenticated User

This endpoint can be called after an engagement has started. It indicates that the user has authenticated and updates the persistentCustomerID of the engagement. The client needs to use this endpoint to update NDEP when a user authenticates on the client mobile app and a chat is currently active.

URI	SetAuthenticatedUser
HTTP Method	POST
Formats	JSON

Parameters

Name	Value	Type	Required
authenticatedUser	JSON object of persistent customer IDs. 1. The authenticatedUser object will consist of persistentCustomerID object which will itself be an object. 2. The authenticatedUser object can have additional fields that are client-specific that can be used to verify the user.	string	no
conversationChannel	Channel via which the conversation was engaged.	string	no
customerID	Nuance customer ID	string	yes
engagementID	Nuance engagement ID	string	yes

Example JSON for authenticatedUser

```
authenicatedUser={
   "persistentCustomerID":{
      "email":"test@nuance.com",
      "phone":"+1321546487"
   },
   "field1":"value1",
   "field2":"value2"
}
```

Survey URL The purpose of the Survey URL call is to provide clients a way to launch a post chat survey from an application that uses the Customer Engagement API. This endpoint operates in two modes:

1. The endpoint allows the user to pass the engagementID and customerID of an active engagement and return the base URL and parameters for launching the native post chat survey configured for the site's agent group.
2. The endpoint allows the user to pass a survey spec ID and return the base URL for the survey configured for the given survey spec ID.

URI	surveyUrl
HTTP Method	GET
Formats	XML or JSON

Parameters

Post-Chat Survey URL of an active engagement

Name	Value	Type	Required
customerID	The customer ID value returned in the engagement request API call	string	yes
engagementID	The engagement ID value returned in the engagement request API call.	string	yes
output	XML or JSON. Default is XML.	string	no
siteID	Site ID to check that Site ID is corresponded for surveySpecID	string	yes

Post-Chat Survey URL from KeySurvey with SurveySpecID

Name	Value	Type	Required
output	XML or JSON. Default is XML.	string	no
siteID	Site ID to check that Site ID is corresponded for surveySpecID	string	yes
surveySpecID	Survey spec ID to get parameters for survey.	string	yes

Implementing Post-Chat Survey
Once the preparations are completed in Portal, the client is ready to implement the post-chat survey functionality from within their CEAPI app.

During an active engagement, the app should request the post-chat survey URL from the Customer Engagement API. This is done by submitting a GET request to the /surveyUrl endpoint. The site id, engagement id, and customer id must also be specified.

Here is an example call:

```
https:// api.touchcommerce.com/engagementAPI/v2/customer/surveyUrl?siteID=10003715&customerID=-
4489709243015478260&engagementID=-4489709243015203473&output=json
```

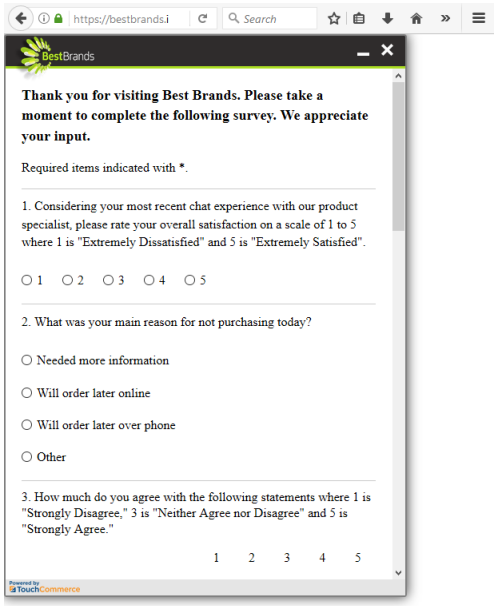
This call will return a URL and a string of parameters similar to the following:

```
{

    "surveyUrl": "https:// bestbrands.inq.com/tagserver/surveys/launchNativePostChatSurvey",

    "surveyLaunchData": "engagementID=-4489709243015203473&customerID=-
4489709243015478260&siteID=10003715&surveyID=2022&agentGroupID=10004026&businessUnitID=13000508&age
ntID=achung_a"

}
```

The survey can be launched by loading a web browser and by either a GET or a POST request. To use a GET request, load the page at the survey URL specifying the launch data as a query string. In the example shown above, the loaded URL will look like this:

```
https:// bestbrands.inq.com/tagserver/surveys/launchNativePostChatSurvey?engagementID=-
4489709243015203473&customerID=-
4489709243015478260&siteID=10003715&surveyID=2022&agentGroupID=10004026&businessUnitID=13000508&age
ntID=achung_a
```

The resulting page will launch the post-chat survey within a browser window.



To integrate this experience within the scope of a native app, it is recommended to open the post-chat survey via Webview client. Normally a Webview window does not display the address bar, which contains engagement specific information.

Alternately, the post-chat survey URL will also accept a POST command with the same parameters. This approach is recommended if there are concerns about the engagement details in the address bar.

Survey URLs

The survey URL that is returned by the **/surveyUrl** request will generally remain constant. However, apps should not have this URL hard-coded, and instead should use the URL that is returned. This prevents future changes to the survey URL from interrupting post-chat survey functionality.

Chat Themes for Post-Chat Surveys

The chat themes that house the post-chat surveys are set within Nuance Portal and are set per device type. The correct theme is loaded automatically when the browser loads the URL. This means that the URL will not change for apps that run on desktop, tablet, or mobile devices.

Note. If you need to deploy new apps, you must notify your Client Services Manager so Nuance can ensure that a device-appropriate chat theme is available for the new app's post-chat survey.

Request Example

https:// api.touchcommerce.com/engagementAPI/v2/customer/surveyUrl?siteID=10003715&customerID=-1655537766535707460&engagementID=-1655537766535432673&output=json

Request Example for KeySurvey Post Chat Survey URL

Request Example

```
GET https://
api.touchcommerce.com/engagementAPI/v2/customer/surveyUrl?siteID=306&surveySpecID=2&output=xml
```

Response Example

```
200 OK

<message><SurveyUrl>http://
www.keysurvey.com/survey/267053/e7f8/?LQID=1&&surveyID=267053</SurveyUrl></message>
```

Topic List (GET)

This endpoint is used to get a list of all topics for the current customer. Filtering is by **customerID** or **persistentCustomerID**. This call is available for asynchronous chats *only*. It cannot be used for non-asynchronous chats

URI	topicList
HTTP Method	GET
Formats	XML or JSON

Parameters

Name	Value	Type	Required
businessUnitID	Business unit ID. (body parameter)	string	no
customerID	The customerID parameter is required for correct balancing. (query parameter)	string	yes
output	JSON or XML. XML is the default	string	no

Name	Value	Type	Required
persistentCustomerID	Private customer identifier from the client side in JSON format. Example: (“email”, “phone”, “123”) **NOTE**: if any value in JSON key-value pair is empty then request will be failed. Following example of invalid persistentCustomerID: {"email":"", "phone":"111-11-11"} If conversation filter setting is not specified then filtering by customerID is used. **IMPORTANT**: When client use conversation filter by persistent customer ID it is important to set **unique** persistentCustomerID per real client (usually when customer logs in set any specific data like ID). Potential consequence of using the same persistent customer ID for different customer is possibility to see data of another customer with the same persistent customer ID even on another device and another customerID. ***Example:*** Here is an example of a bad persistentCustomerID persistentCustomerID={"name":"John"} In this case every customer with the name John will be able to see data from the others. Best practice to use unique ID of customer based on login data: persistentCustomerID={"id":"123484398578395389457938475983473949590"}	string	no
siteID	Used to check that site ID is corresponded for the user.	string	yes

Update Device Id

This endpoint is used to update the device id for push notification functionality for a conversation in case of absent active engagement.

URI	updateDeviceId
HTTP Method	POST
Formats	XML or JSON

Parameters

Name	Value	Type	Required
businessUnitID	Business unit ID. (body parameter)	string	no
customerID	The customerID parameter is required for correct balancing. (query parameter)	string	yes
pushNotificationDeviceID	Device ID for push notification functionality. (body parameter) **Note.** You can remove pushNotificationDeviceId values using an empty pushNotificationDeviceId call.	string	yes
siteID	Possible values: "closed". (body parameter)	string	yes

Note. If the client uses OAuth authorization, then valid access token should be sent in the “Authorization” header. Otherwise the client should log into the API as engagement API user.

Example of Sending Message

POST `https://api.touchcommerce.com/engagementAPI/v2/customer/updateDeviceId`

```
POST body:

customerID=-7599162224901667826&businessUnitID=22&pushNotificationDeviceID=1&siteID=306
```

Response Example

```
200 OK
```

Verify Conversation Token (POST)

This endpoint is used to verify the conversation token generated for cross-channel chats by agent and return the conversationObjectID and the token in JSON.

This is only for asynchronous chat users.

URI	verifyToken
HTTP Method	POST
Formats	JSON

Parameters

Name	Value	Type	Required
customerID	The customer ID value returned in the engagement request API call.	string	yes
conversationTokenID	The token ID value passed from the chat landing page for cross-channel hopping.	string	yes
siteID	The site ID to check that the site ID corresponds to the user.		

Example of Sending Message

```
GET
https://api.touchcommerce.com/engagementAPI/v2/customer/verifyToken?siteID=306&customerID=1234567&conversationTokenID=1232134324
```

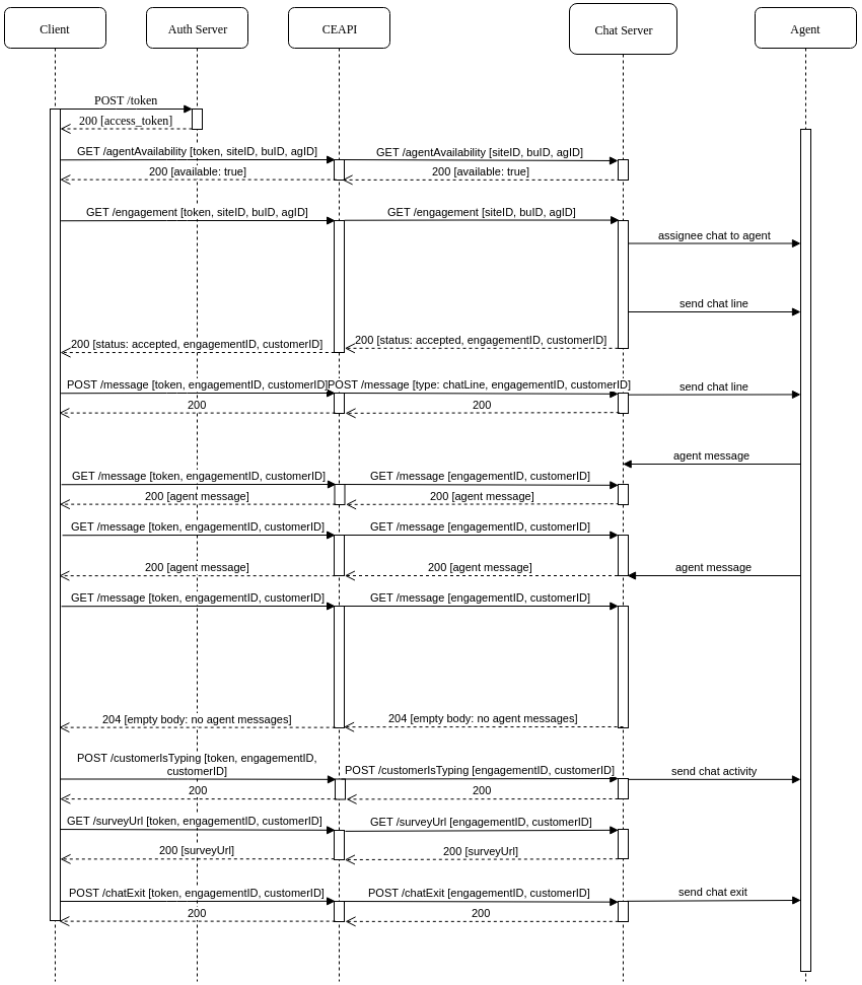
Response Example

```
200 OK

Body: {"conversationTokenID": "1232134324","conversationObjectID": "a634mf7f9a9")
```

HTTP Response Codes

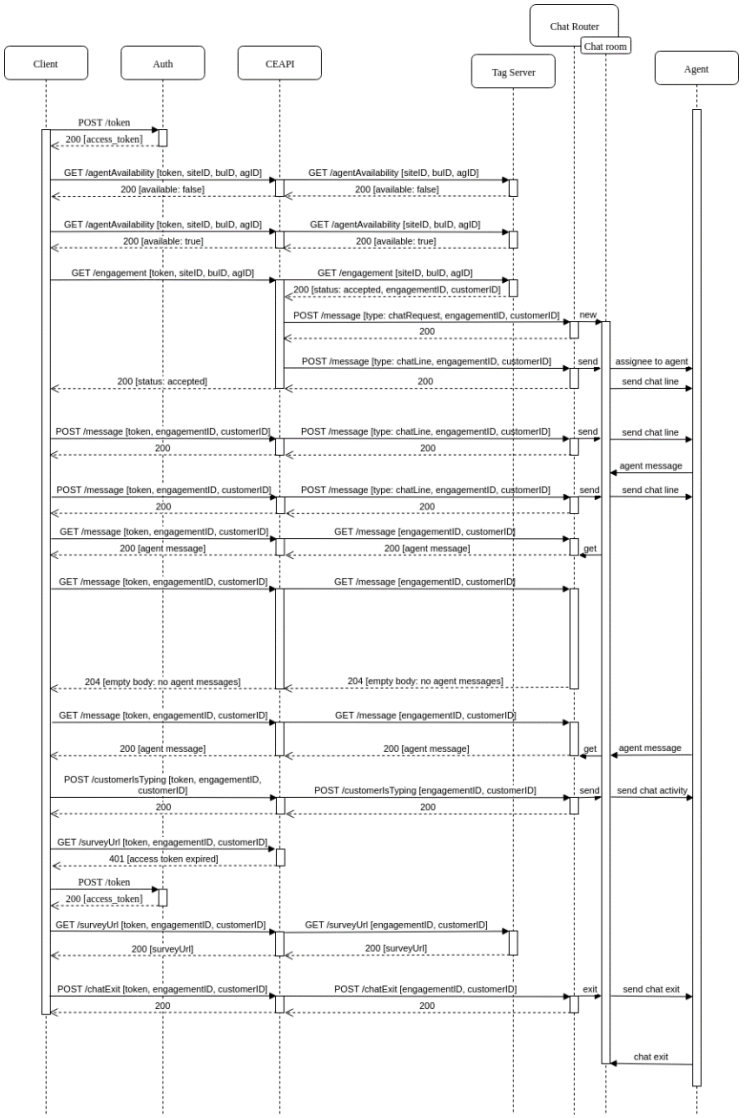
Code	Meaning
200	Message was accepted by server.
400	Request has incorrect parameters
401	• Invalid clientID and/or invalid clientSecret • Invalid grant type is presented as part of a POST Oath-2 request • API call presented an expired access token
403	User account does not have privileges to use the API or to see the requested site.
500	Server error

CEAPI Sequence Diagram with Long Polling Flow (Customer experience diagram)

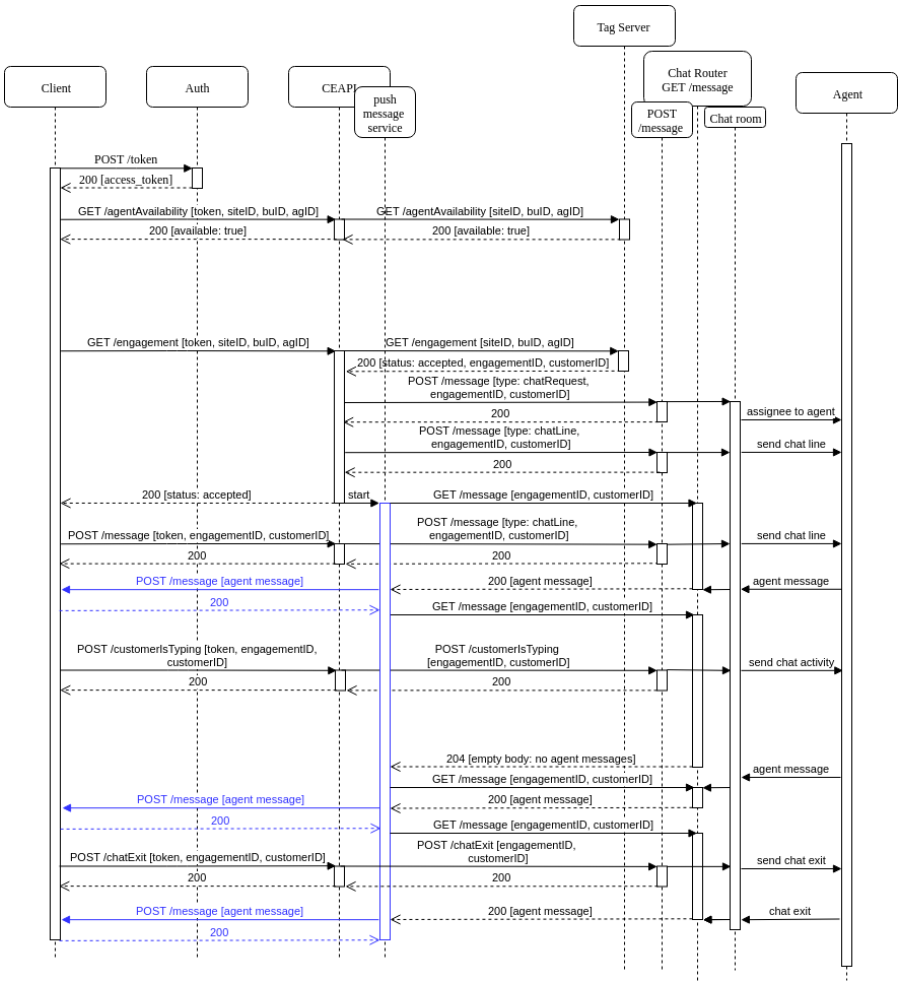


NUANCE

Detailed diagram

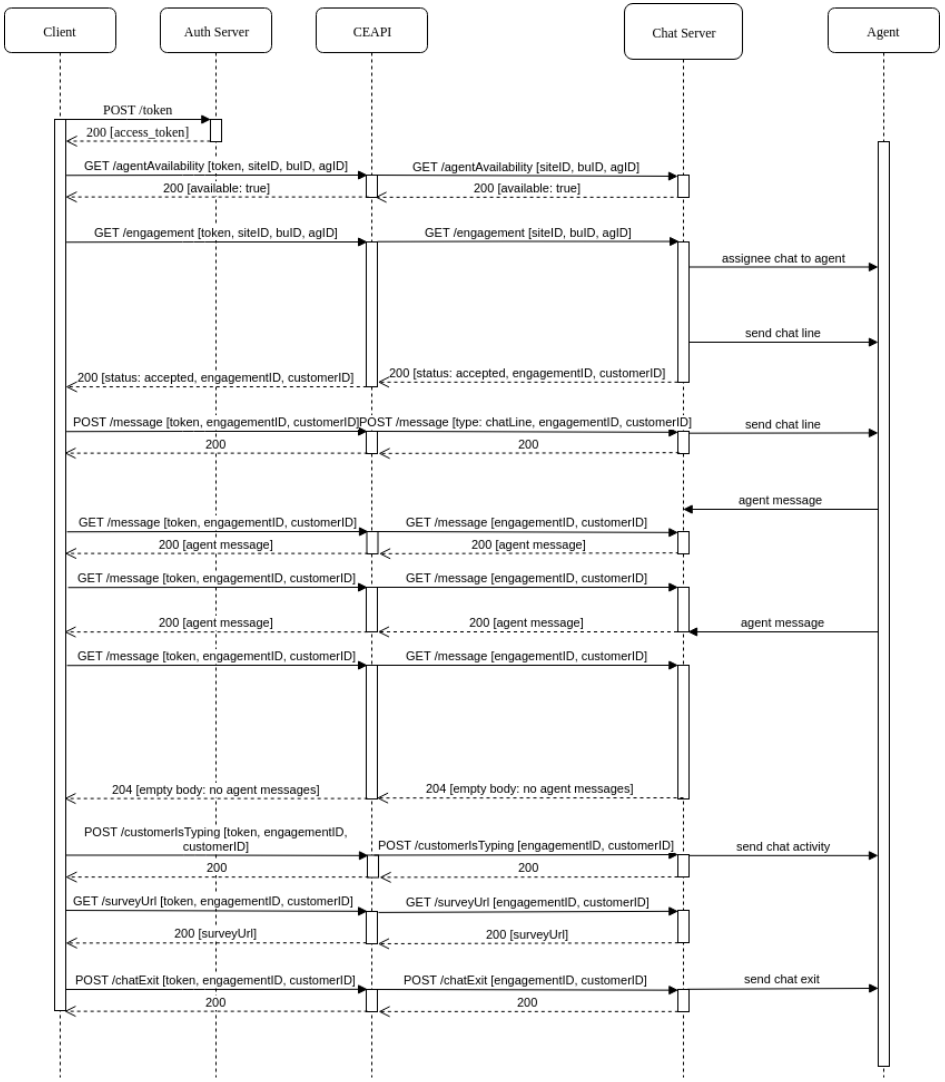


CEAPI with POST Agent Messages Flow (Customer experience diagram)



NUANCE

CEAPI Diagram with Long Polling Flow



NUANCE