Journey JavaScript SDK

9/18/2024

Table of Contents

Contents

Learn how to use the Journey JavaScript SDK to customize how Genesys Predictive Engagement tracks and manages customer activity on your website.

> **Important**
> The articles in this guide only apply to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Get started

Learn how to get started using the Journey JavaScript SDK.

- Get started
- About the tracking snippet
- Cookies
- Advanced tracking with cookies

Methods

Learn about the methods that are available in the Journey JavaScript SDK.

- Method reference
- Initialization methods
- Session methods
- Tracking methods
- Event methods
- Utility methods

APIs

Learn how to use APIs to track user activity, form submissions, and abandonment events.

- Web Tracking API
- Form Tracking API

Trait mapping

Learn how to map multiple records for the same customer to see more complete customer profiles.

- Map traits to link customer records

Modules

Learn about JavaScript files that enhance the functionality that the Journey JavaScript SDK provides.

- About modules

Get started

Get started using the Journey JavaScript SDK.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

When you deploy the Genesys Predictive Engagement tracking snippet on your website, you initialize the Journey JavaScript SDK. You can enhance and refine the tracking snippet using the Journey JavaScript SDK.

- Copy and deploy the tracking snippet.
- Track activity in a GDPR-compliant manner.
- Configure advanced tracking.
- Understand how Genesys Predictive Engagement uses cookies.
- Use the Web Tracking API.
- Method reference

About the tracking snippet

Contents

Learn what happens after you deploy the tracking snippet on your webpages.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

How the tracking snippet works

Genesys Predictive Engagement provides a traditional tracking snippet and an SPA tracking snippet to track activity on your webpages. For more information, see Types of snippets.

When you add the snippet to a webpage, the tracking snippet loads the Journey JavaScript SDK whenever a visitor accesses a tracked page. To ensure that the process of loading the Journey JavaScript SDK does not cause the visitor to wait for the page to load, we cache the Journey JavaScript SDK in the visitor's browser and load it asynchronously.

> **Important**
> - The Genesys Predictive Engagement tracking snippet loads JavaScript asynchronously without slowing down page loading.
> - The Genesys Predictive Engagement SDK does not block the loading of any other resources.
> - This snippet represents the minimum configuration needed to add the Genesys Predictive Engagement customer support widget. To start sending tracking data from your visitors back to the Genesys Predictive Engagement servers, see Web Tracking API.

Once loaded, the Journey JavaScript SDK:

- Creates a new script HTML element
- Sets the source attribute to the Genesys Predictive Engagement SDK’s URL
- Sets the async attribute to 1 (truthy)
- Adds the script element to the DOM
- Sets the name of the only global function exposed by the Genesys Predictive Engagement SDK to `'ac'`
- Calls the ac function and runs the following commands:

- `init` to set the organization ID and region, and to specify which account to send the data to
- `pageview` to record an event when a page with the tracking snippet loads, allowing Genesys Predictive Engagement to track the visitor's journey across the website.

For more information about the the tracking method, see Web Tracking API.

Cookies

Contents

Configure how Genesys Predictive Engagement uses cookies to store customer data.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Purpose

Genesys Predictive Engagement uses cookies to store non-sensitive data in the browser. The visitor's browser must allow cookies for Genesys Predictive Engagement to work properly.

The cookie that identifies customers

To identify customers, Genesys Predictive Engagement uses a first-party cookie named **_actmu** to store the visitor ID. This cookie is a unique, randomly generated string that is stored in the browser. This cookie is sent to the Genesys Predictive Engagement APIs to determine whether a **tracked event** is associated with a particular customer and to associate subsequent visits to the same site with the same customer.

Cookies that expire after 1 year

> **Important**
> You can change the expiration time for all cookies that expire after 1 year. For more information, see Advanced tracking with cookies.

> **Important**

> On January 31, 2023, Genesys removed the functionality of the identify method that is used to add a customer record. After the removal date, the _actmi and _actmh cookies will no longer be created or updated. The existing cookies will retain the information that they had before the removal date.

Cookie name	Purpose
_actcc	Distinguishes visitor's `beacon` and `pageview` counts for the current session and all sessions collectively.
_actmi	Distinguishes logged in visitors. This cookie is set to the user ID passed when calling the `identify` method.
_actmu	Distinguishes visitors. The cookie is created when the Journey JavaScript SDK library executes and no existing __actmu cookies exists.
_actvc	Distinguishes the visit count for an individual visitor. This cookie is created and updated on each separate visit.
_actts	Distinguishes timestamps of the visitor's first, previous, and current session.

Cookies that expire after 30 minutes

Cookie name	Purpose
_actmm	Distinguishes UTM information.
_actmr	Distinguishes the session referrer.
_actms	Distinguishes session ID.

Cookies that expire at varying times

Cookie name	Purpose	Expiration details
_ac_test	Genesys Predictive Engagement uses this cookie to check whether the browser supports first-party cookies and whether Genesys Predictive Engagement can set the tracking cookies successfully.	Immediately after it is set.
_actmf	Stores data submitted in a form and sends it on the next page load.	If this cookie is not set before the visitor leaves the site, the cookie expires when the session expires.
_actmh	Stores a hash of the visitor	

	information that is passed when calling `identify` to minimize the number the times that this information is sent to the Genesys Predictive Engagement servers.	

Advanced tracking with cookies

Contents

Learn how to use the Journey JavaScript SDK to refine how Genesys Predictive Engagement tracks visitor data. Alternatively, you can refine tracking using your preferred tag manager. For more information, see About event tracking with tag managers.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

About advanced tracking

Use `init` options to change how Genesys Predictive Engagement sets cookies and how you track your visitors across subdomains or multiple domains.

Cookie options

By default, the Journey JavaScript SDK sets the cookie expiration date and determines the cookie domain. To customize these settings, use the following parameters with the `init` method.

me	Description	Default
`allowedLinkers`	Array of domains that can link into the current domain for cross-domain tracking.	Null
`autoLink`	Augments all links to the specified domains on the site to contain information that allows the linked page to continue the current tracking session.	Null
`cookieDomain`	Determines the domain on which the cookies are set.	The highest level domain possible
`cookieExpires`	Specifies the expiration time in seconds for the `actmi`, `_actmu` and `_actvc` cookies. For example, 1 year = 365 days * 24 hours * 60 minutes * 60 seconds = 31536000 seconds.	1 year
`cookiePrefix`	Adds a prefix to the names of the Genesys Predictive Engagement cookies.	"_" (underscore)

Example

```
ac('init', 'YOUR-ORGANIZATION-ID', {
  region: 'YOUR-REGION',
  cookieDomain: 'YOUR-DOMAIN',
  cookieExpires: 31536000,
  cookiePrefix: 'YOUR-PREFIX'
});
```

Track a domain and its subdomains

By default, to simplify cross-domain tracking implementations, Genesys Predictive Engagement writes cookies to the highest level domain possible. If you manage both a domain and one or more subdomains such as `www.example.com, blog.example.com` and `store.example.com`, the cookie domain used to store cookies will be `.example.com`.

Tracking subdomains in 2 different accounts

For tracking subdomains separately in 2 different accounts, you need to customize the Genesys Predictive Engagement tracking snippet to specify the desired domain in the `init` call:

```
ac('init', 'YOUR-ORGANIZATION-ID', {
  region: 'YOUR-REGION',
  cookieDomain: 'subdomain.example.co.uk'
});
```

Track multiple domains

A default setup tracks traffic to each domain (for example: `example.com` and `example.co.uk`) independently. Therefore, a visitor arriving in one domain who then proceeds to another domain that is set up with the same tracking account counts as two separate visitors with two separate visits or sessions. Each one comprises the activities (pages visited, and so on) that occurred on each domain.

To enable cross domain tracking, also known as *site linking*, to bundle together the traffic to both domains, modify the `init` call in the Genesys Predictive Engagement tracking snippet to allow auto linking to another domain. Suppose you have a site, `example.com` that links to `example.co.uk` and in opposite manner.

To enable tracking across both of these domains, modify the following in the Genesys Predictive Engagement tracking snippet in `example.com`:

```
ac('init', 'YOUR-ORGANIZATION-ID', {
  region: 'YOUR-REGION',
  allowedLinkers: ['example.co.uk'],
  autoLink: ['example.co.uk']
});
```

Then, modify the Genesys Predictive Engagement tracking snippet in `example.co.uk` to also allow

linking. Specifically, to accept the visitor tracking cookies from another site, and to enable auto linking to decorate all links pointing to example.com:

```
ac('init', 'YOUR-ORGANIZATION-ID', {
  region: 'YOUR-REGION',
  allowedLinkers: ['example.com'],
  autoLink: ['example.com']
});
```

Asymmetric site linking

You can set up asymmetric site linking. It allows you to carry cookies from `example.com` to `example.co.uk` but not the other way around. The Genesys Predictive Engagement tracking snippet in `example.com` removes `example.co.uk` from `allowedLinkers`, and removes `autoLink` from the Genesys Predictive Engagement tracking snippet added to `example.co.uk`.

Method reference

Contents

View a list of methods available in the Journey JavaScript SDK.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Initialization methods

For more information about a method, including signature and arguments, click the method's name.

Method	Description
init	Initializes the Journey JavaScript SDK.
initialized	Notifies when the Journey JavaScript SDK is initialized fully.
destroy	Stops all Journey JavaScript SDK activity and removes all tracking information.

Session methods

For more information about a method, including signature and arguments, click the method's name.

Method	Description
api.session.getData	Returns an object that contains session information.
api.session.getCustomerCookieId	Returns a string that contains the customer cookie ID.
api.session.getId	Returns a string containing session ID.

Tracking methods

For more information about a method, including signature and arguments, click the method's name.

Method	Description
pageview	Tracks page views.
record	Records custom website events.
forms:track	Captures when visitors complete web-based forms.

Events methods

For more information about a method, including signature and arguments, click the method's name.

Method	Description
on	Subscribes to notifications about a particular type of Journey JavaScript SDK activity.
once	Subscribes to notifications for first event for a particular type of Journey JavaScript SDK activity.
off	Unsubscribes from notifications about a particular type of Journey JavaScript SDK activity.

Utility methods

For more information about a method, click the method's name.

Method	Description
serialize	Passes serialized data from a form.
dom - ready	Allows you to specify a function handler to run when the DOM has loaded fully.
debug	Sends helpful messages to the console.

Initialization methods

Contents

View the methods that initialize and destroy the Journey JavaScript SDK.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Initialization methods

- init
- initialized
- destroy

init

Contents

Learn how to use the `init` method to initialize the Journey JavaScript SDK.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `init` method initializes the Journey JavaScript SDK.

> **Important**
> For GDPR compliance, obtain a customer's consent before you call the `init` method. For more information about how to use Genesys Predictive Engagement in a GDPR-compliant manner, see GDPR.

Signature

```
ac('init', organisationId, options)
```

Arguments

- organizationId
- options

organizationId

- **Description:** Your organization's unique ID. Find your organization ID
- **Type:** String
- **Status:** Required

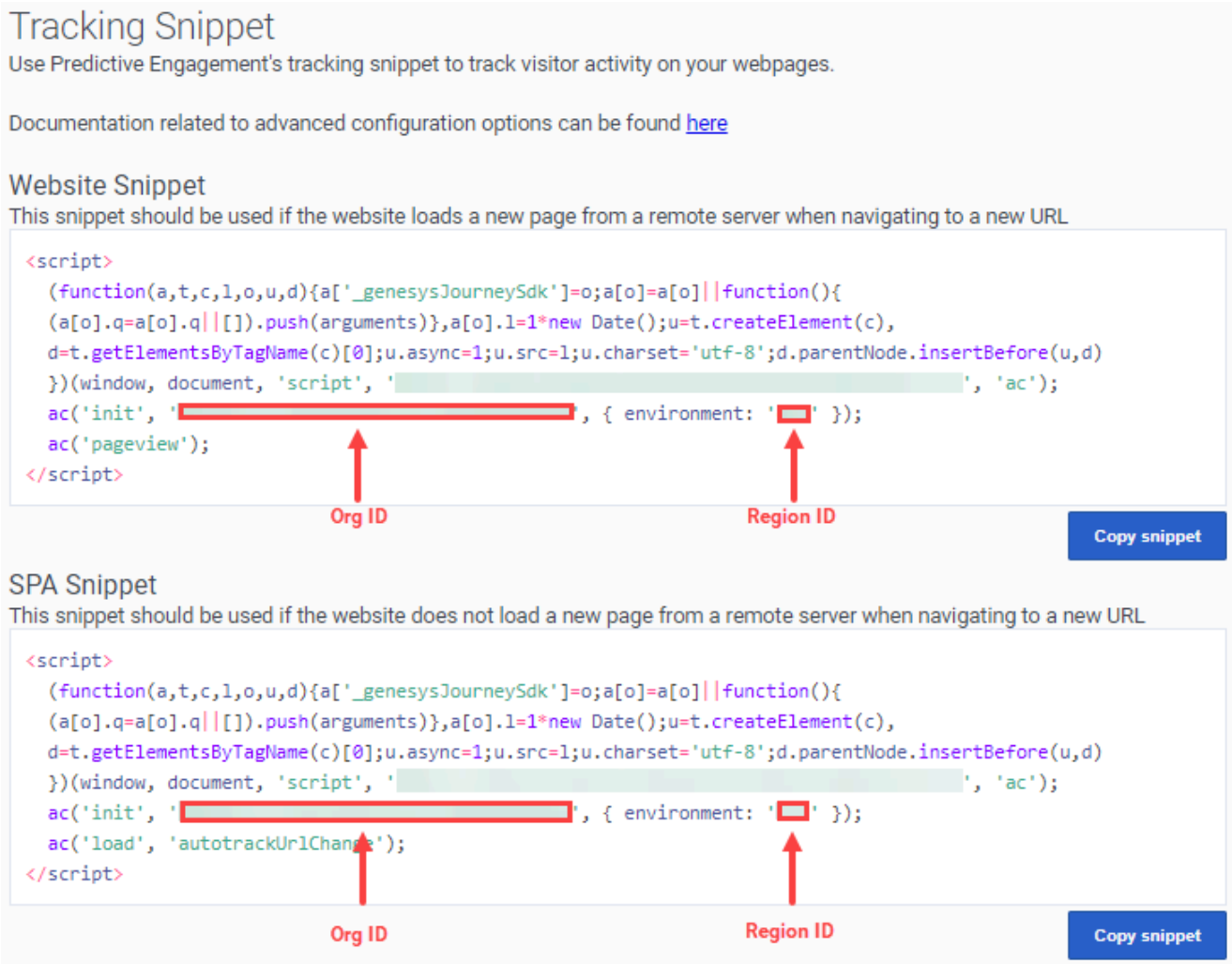
options

> **Tip**
> For detailed explanations of how you can use these options to configure tracking, see Advanced tracking with cookies.

- **Description:** Configures the Journey JavaScript SDK with its region and other known default options.
- **Type:** Object
- **Status:** Required
- **Properties:** See the following table.

Name	Description
region	Your organization's region. Find your region ID
cookieDomain	Sets a custom cookie domain.
cookieExpires	Sets a time in seconds for a visitor's cookie to expire.
cookiePrefix	Sets a custom cookie prefix.
autoLink	An array of domains whose outbound links will be modified to contain tracking infor advanced tracking.
allowedLinkers	An array of domains whose inbound referral will set tracking information. For an exa
canonicalLink	Uses canonical links.
globalTraitsMapper	Maps custom attributes to traits. For more information, see Traits Mapper.

Find your org ID and region ID



Go to **Genesys Cloud CX** > **Admin** > **Tracking Snippet**.

Region names and IDs

The following table lists the available region names and corresponding IDs.

Region name	ID
Americas (US West)	usw2
Americas (US East)	use1
Americas (Canada)	cac1
EMEA (Dublin)	euw1
EMEA (London)	euw2

Region name	ID
EMEA (Frankfurt)	euc1
Asia Pacific (Tokyo)	apne1
Asia Pacific (Seoul)	apne2
Asia Pacific (Sydney)	apse2

initialized

Contents

Learn how to use the `initialized` method to receive notification when the Journey JavaScript SDK has initialized fully.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

Use the `initialized` method to receive notification when the Journey JavaScript SDK initializes fully. This information is useful in situations that require tracking data from the Journey JavaScript SDK, but because SDK initialization takes place after the page loads, tracking has not begun.

For example, some businesses require GDPR consent before they begin tracking visitor activity. A business can present a GDPR consent confirmation request to a visitor when the visitor arrives at the webpage. Until the visitor agrees to allow tracking, the Journey JavaScript SDK remains in an uninitialized state, and calls to SDK methods, such as the api.session methods, fail.

After the visitor provides their consent, the SDK initializes and can begin tracking visitor activity. Once initialized, callbacks registered using the initialized method are invoked and can begin to use the Journey JavaScript SDK's other methods.

Signature

```
ac('initialized',  eventHandler);
```

Example

```
ac('initialized', () => {
  console.log('Tracking SDK initialized');
  ac('api.session.getData', (session) => {
    console.log('Session data', session);
  });
});
```

destroy

Contents

Learn how to use the `destroy` method to stop all Journey JavaScript SDK activity and remove all tracking information.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `destroy` method stops all Genesys Predictive Engagement SDK activity and removes all tracking information.

Signature

```
ac('destroy')
```

Arguments

None.

GDPR and destroyed data

The `destroy` method stops all Genesys Predictive Engagement tracking immediately. It also removes all cookies that the SDK set; and doesn't offer anymore proactive engagements. Use the `destroy` method when a customer requests that you stop tracking their activities on your website. For more information about using Genesys Predictive Engagement in a GDPR-compliant manner, see GDPR.

Session methods

Contents

View the methods that allow you to obtain data about a specific session.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Session methods

Genesys Predictive Engagement session methods allow you to obtain session-specific data from web sessions.

- api.session.getCustomerCookieId
- api.session.getData
- api.session.getId

api.session.getCustomerCookieId

Contents

Learn how to obtain a customer's cookie ID.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

api.session.getCustomerCookieId returns a string that contains the customer's cookie ID.

Signature

```
ac('api.session.getCustomerCookieId', (err, cookieInfo) => {
  if (err) {
    // handle error
    return;
  }

  return cookieInfo;
});
```

Callback

The callback takes err as the first parameter.

api.session.getData

Contents

Learn how to obtain the data such as the short ID for a particular customer's session.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

api.session.getData returns an object that contains the session ID, short ID, and customer cookie ID.

Signature

```
ac('api.session.getData', (err, sessionInfo) => {
  if (err) {
    // handle error
    return;
  }

  return sessionInfo;
});
```

Example

The following is an example of an object that api.session.getData returns:

```
{
  id:
  shortId: 12345
  customerCookieId:
}
```

Callback

The callback takes err as the first parameter.

api.session.getId

Contents

Learn how to get the Id of a particular session.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

api.session.getData returns an object that contains session ID.

Signature

```
ac('api.session.getId', (err, IdInfo) => {
  if (err) {
    // handle error
    return;
  }

  return IdInfo;
});
```

Callback

The callback takes err as the first parameter.

Tracking methods

Contents

Learn about the methods available in the Journey JavaScript SDK that you can use to extend and enhance how Genesys Predictive Engagement tracks visitor activity on your website.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Tracking methods

- pageview
- record
- identify
- forms:track

Events that these methods track are custom web events.

pageview

Contents

Learn how to use the `pageview` method to track when visitors view your webpages.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `Pageview` method tracks page views.

Signature

```
ac('pageview', [pageOverrides], [customAttributes], [options])
```

Arguments

- pageOverrides
- customAttributes
- options

pageOverrides

- **Description:** Sets custom page view location and title
- **Type:** Object
- **Status:** Optional
- **Properties:** See the following table.

Name	Description	Type	Status	Default
location	Page URL	string	optional	
title	Page title	string	optional	

customAttributes

- **Description:** Adds extra information to `pageview` event
- **Type:** Object
- **Status:** Optional
- **Restrictions:** Object with properties of type string, number, Boolean or customAttribute. See customAttributes.

options

- **Description:** Use for more configuration
- **Type:** Object
- **Status:** Optional
- **Properties:** See the following table.

Name	Description	Type	Status	Default
traitsMapper	Used to map custom attributes to traits. For more information, see Traits Mapper.	traitsMapper		
callback	Called once beacon is sent	function	optional	
callbackTimeout	ms to wait for beacon to send	number	optional	

record

Contents

Learn how to use the `record` method to capture website events.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `record` method records custom website events.

Signature

```
ac('record', eventName, [customAttributes], [options])
```

Arguments

- eventName
- customAttributes
- options

eventName

- **Description:** Name of the custom event
- **Type:** String
- **Status:** Required
- **Restrictions**: See Guidelines for custom event names.

customAttributes

- **Description:** Adds extra information to `pageview` event
- **Type:** Object
- **Status:** Optional
- **Restrictions:** Flat object with properties of type string, number, Boolean, or customAttribute

Example

```
ac('record', 'product_added', { price: 15.99, code: 'CDE-123', name: 'Product',
hasBatteries: false });
```

Additionally, define the datatype for the attribute to define the value better.

Example

```
ac('record', 'product_added', { price: {datatype: 'integer' value: 15, name:
{datatype: 'string' value: 'Product'}}});
```

customAttributes for outcome value tracker

To derive the value of an outcome, create an event that comprises an associated value field. Then, using this event and the value field (within the same event), track the value of the outcome you created.

For example, the following event tracks the products added with the value, price. When you create an outcome using this event, the products added along with their value is tracked as part of the outcome.

- **Description:** Adds extra information to `pageview` event
- **Type:** Object
- **Status:** Optional
- **Restrictions:** Flat object with properties of type string, number, or Boolean

Example

```
ac('record', 'product_added', { price: 15, code: 'CDE-123', name: 'Product',
hasBatteries: false });
```

options

- **Description:** Used for more configuration

- **Type:** Object
- **Status:** Optional
- **Properties:** See the following table.

Name	Description	Type	Status	Default
traitsMapper	Used to map custom attributes to traits. For more information, see Traits Mapper.	traitsMapper		
callback	Called once beacon is sent	function	optional	
callbackTimeout	ms to wait for beacon to send	number	optional	

identify

Contents

Learn how to use the `identify` method to add information to a customer record.

> **Important**
> On January 31, 2023, Genesys removed the functionality of the identify method that is used to add a customer record. For more information, see Deprecation: identify journey SDK method.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `identify` method adds information to a customer record.

> **Important**
> `identify` affects the next web event sent using the `pageview` or `record` method.

Signature

```
ac('identify', [loginId], [traits], [callback])
```

Arguments

- loginId
- traits

- callback

login

- **Description:** ID used to stitch customer identities together
- **Type:** String/null
- **Status:** Required
- **Note:** If loginId is null, the current identity is cleared.

traits

- **Description:** Information about a customer
- **Type:** Object
- **Status:** Optional
- **Restrictions:** Flat object with properties of type string, number, or Boolean

For more information about how to link different customer records, see Traits mapper.

callback

- **Description:** Callback
- **Type:** Function
- **Status:** Optional

forms:track

Contents

Learn how to use the`forms:track` method to capture when visitors complete web-based forms.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `forms:track` method tracks form submission and abandonment events. By default, forms tracking captures form data when a visitor submits or abandons a form.

- Recorded form data **includes** the values of all input, select, and text area fields.
- Recorded form data **excludes** the values of hidden, submit, and password fields, along with any fields that contain any of the sensitive input strings.

> **Important**
> For Genesys Predictive Engagement SDK forms tracking to capture form data, each input requires a properly defined `name` attribute.

For more information, see Form Tracking API.

Signature

```
ac('forms:track', [selector], [options]);
```

Arguments

- selector

- options

selector

- **Description:** CSS selector for the element or elements to track
- **Type:** String
- **Status:** Optional
- **Default:** Form

options

- **Description:** Activity or behavior to track
- **Type:** Object
- **Status:** Optional
- **Default:** {}
- **Properties:** See the following table.

Name	Description	Type	Status	Default	Arguments
captureFormDataOnAbandon		Boolean	optional	true	
captureFormDataOnSubmit		Boolean	optional	true	
transform		function	optional		formDataObject
traitsMapper		traitsMapper			
customAttributes	Field used to send additional information when a form-related event occurs. The field can be set with static, predetermined values. Once the form tracker module initiates, the value that is available in the value field is applied for the rest of the session. The value cannot be changed	object	optional		

	dynamically according to user action. There can be more than one field.				

Example

To create an event to track the number of customers who sign themselves up from the sign-up page, use the customAttributes within the form:track SDK as follows:

```
{
captureFormDataOnSubmit: [
'forms:track', 'sign-up-form', { captureFormOnSubmit: true, customAttributes: {formValue: 200}
]
}
```

Use events to track outcome value

Use the attributes from the form:track SDK to track an outcome from action maps. You can further use the value set within the SDK to the define the value of the outcome. In this example, the value of sign up is set at 200. This means that if the outcome value of the outcome stands at 2000, the total number of sign ups is 10 whose with a total value of 2000.

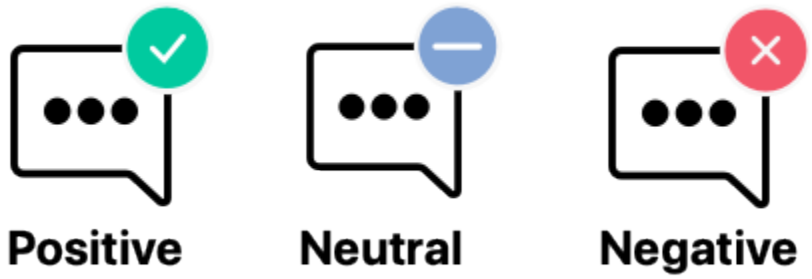
Display icons in the Journey gadget

Contents

Learn how to use the SDK to display icons for tracked visitor behavior on the Journey map.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

About the icons

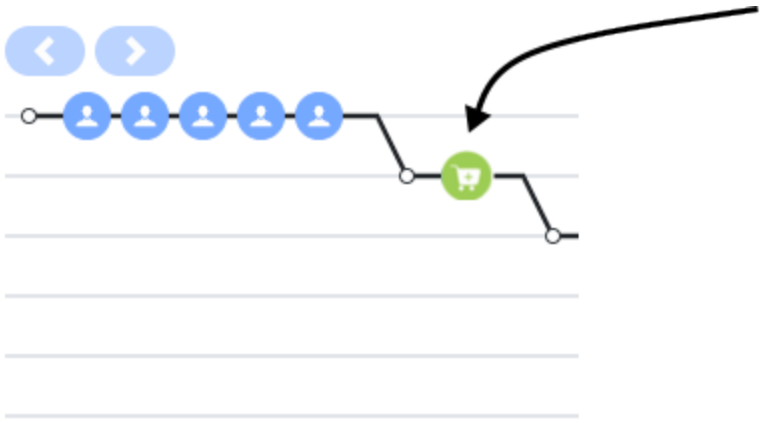


Use the `ac('record')` method to display a Genesys Predictive Engagement icon on the Visit journey map (admin view) when a visitor completes a tracked behavior.

Available icons

- Purchase-related icons
- Form-related icons
- Journey-related icons

Code example



This code example shows how to use `ac('record')` to display the Product added icon in the Visit journey map (admin view) when a visitor adds a t-shirt to their shopping cart.

```
ac('record', 'product_added', [optionalExtraDataObject]
ac('record', 'product_added', { name: 't-shirt', id: 'hkds9d8j', price: '$45.45' });
```

Purchase-related icons

Icon	Tooltip text	Description	Name
	Product added to cart	Visitor added a product to their shopping cart.	"product_added"
	Product removed from cart	Visitor removed a product from their shopping cart.	"product_removed"
	Checkout complete	Visitor completed the purchase of the items in their shopping cart.	"product_purchased"

Form-related icons

> **Important**

> You can display the icons in this section using `ac('record')` or auto form tracking.

Icon	Tooltip text	Description	Name
	Form submitted	Visitor submitted a form.	"form_submitted"
	Form abandoned	Visitor navigated away from a form before completing it.	"form_abandoned"

Miscellaneous icons

Icon	Tooltip text	Description	Name
	Searched	Visitor searched for the string shown in the tooltip.	"search_performed"

About Events methods

Events methods allow developers to build functionality that reacts to state changes in the SDK.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Contents

Events methods

Events methods allow you to receive notifications about activities that the Journey JavaScript SDK tracks.

- on
- once
- off

Use Events methods with content offers

You can use Events methods to capture data related to Content offers.

- Use Events methods with web actions
- Content offers lifecycle
- Examples: Events methods with content offers

Use Events methods with web chats

You can use Events methods to capture data related to web chats.

- Use Events methods with web actions
- Content offers lifecycle
- Examples: Events methods with web chats

on

Contents

To subscribe to receive notifications about a particular type of SDK activity, use ac('on').

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

Use ac('on') to subscribe to SDK events of a given event type and state.

Signature

```
ac('on', '', function eventHandler(evt) { /* do stuff */ });
```

Examples

For examples, see:

- Examples: Events methods with content offers
- Examples: Events methods with web chats

once

Contents

To receive a notification for only the first SDK event for a particular type of activity, use ac('once').

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

Use ac('once') to subscribe to the first SDK event of a given event type and state.

Signature

```
ac('once', '', function eventHandler(evt) { /* do stuff */ });
```

Example

See:

- Examples: Events methods with content offers
- Examples: Events methods with web chats

off

Contents

To unsubscribe from receiving notifications about a particular type of SDK activity, use ac('off').

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

Use ac('off') to unsubscribe from receiving future SDK events of a given event type and state.

Signature

```
ac('off', '', function eventHandler(evt) { /* do stuff */ });
```

Example

See:

- Examples: Events methods with content offers
- Examples: Events methods with web chats

Use Events methods with web actions

Contents

Use Events methods to subscribe to events that occur during the lifecycle of Genesys Predictive Engagement web actions such as web chats and content offers. This raw data can be streamed to third-party analytics platforms or to tag management platform data layers for use in analytics and reporting platforms.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Event methods for web actions

To capture information about events that occur during the lifecycle of a web action, use the Events methods shown in the following table.

Method	Description
on	Subscribe to start receiving events for a given type of web action in a given state.
once	Receive events for only the first occurrence of a given type of web action in a given state.
off	Unsubscribe to stop receiving events for a given type of web action in a given state.

Media types, lifecycle states, and code examples using Events methods

Media type	Lifecycle states	Code examples
contentoffer	Content offers lifecycle	Examples: Events methods with content offers
webchat	Web chat lifecycle	Examples: Events methods with web chats

Event types for web actions

The following table lists the events that you can use with Events methods for web actions. Event information returned includes the action state, customer ID, session ID, and action map ID.

Event	Data type	Description
actionId	UUIDv4	Unique Id for a specific Predictive Engagement action.
actionState	String	Current state of the action. For example, offered.
actionMediaType	String	The engagement type. For example, webchat.
actionMapId	UUIDv4	Id of the action map that qualified/triggered this action.
actionMapVersion	Integer	Version of the action map.
customerId	UUIDv4	Stable identifier of the customer. For example, a cookie Id.
customerIdType	String	The specific type of customer identifier (always "cookie").
sessionId	UUIDv4	Identifier of the customer's current web session.
errorCode	Integer	Status code for any exceptions caught during presentation of the action.
errorMessage	String	Error message for any exceptions caught during presentation of the action.

Capture more data with Genesys widgets

You can use the Web Action Events API with Genesys Widgets commands to enrich events with more data that may be useful.

For example, the Genesys Cloud CX `conversationId` may be useful in an analytics context. For more information on the Widgets API, see API Commands.

```
ac('on', 'webchat:all', (evt) => {
  _genesys.widgets.bus.command('WebChatService.getSessionData').then((data) => {
    if (data.conversationId) {
      evt.conversationId = data.conversationId;
    }

    someAnalyticsProvider.send(evt);
  })
});
```

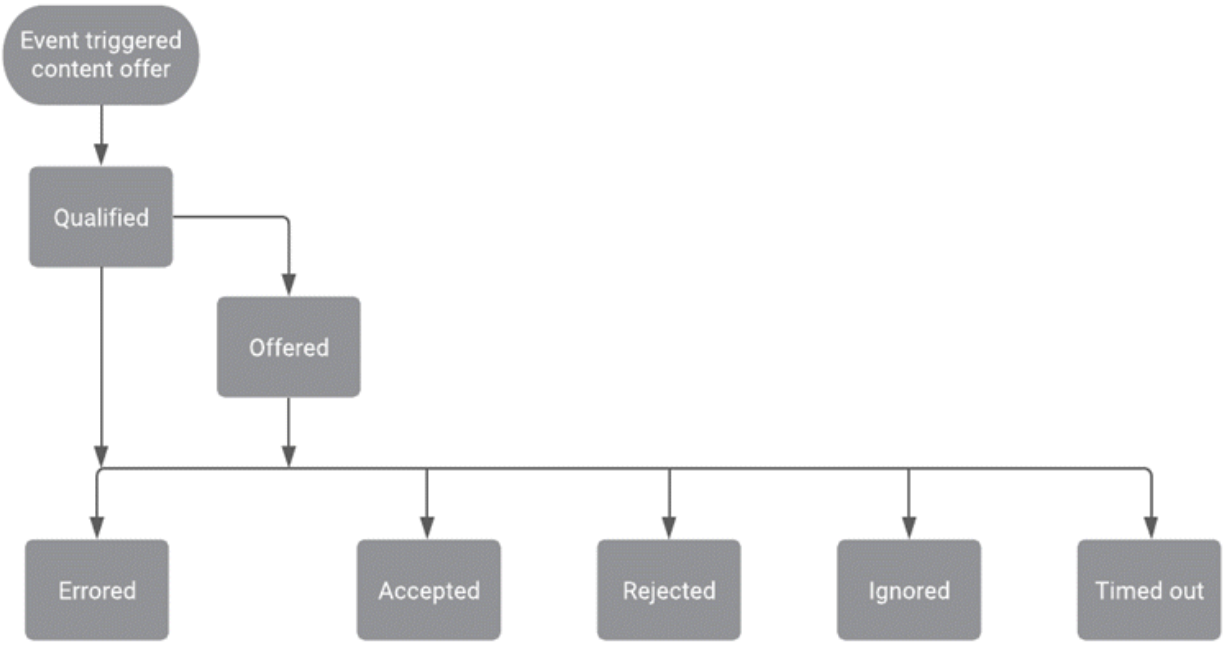
> **Important**
>
> The data that WebChatService.getSessionData returns differs based on your Genesys platform.

Content offers lifecycle

Contents

See the lifecycle of a content offer and the metrics that we capture at each state along the way. We use lifecycle states in reporting and when determining the triggering behavior of action maps that use content offers.

State transitions for content offers



Content offer lifecycle states

The following table provides details about the lifecycle of a content offer, including the events that can occur and the data that is available for use with the Events methods.

State	Event	Description	Data available
offered	Web Actions Offered	Visitor's activity qualified an action map, and a proactive invitation is offered.	See Events methods with web actions.
accepted	Web Actions Accepted	Visitor accepts the invitation by clicking a button like **Book now!** This state is a terminal state.	See Events methods with web actions.
rejected	Web Actions Rejected	Visitor rejects the invitation by either clicking **X** or a button like, **No, but thank you**. This state is a terminal state.	See Events methods with web actions.
errored	Web Actions Errored	An error occurred in the widget that prevented the engagement from occurring. **Note:** This event does not have a corresponding metric in the Action Map Performance Report.	See Events methods with web actions. Also, the errorMessage field is available.
ignored	Web Actions Ignored	Visitor ignored the invitation by navigating	See Events methods with web actions.

State	Event	Description	Data available
		away from or around it. This state is a terminal state. **Note:** This event does not have a corresponding metric in the Action Map Performance Report.	

Terminal states for content offers

In the content offer lifecycle, certain states are considered *terminal*, or final states. If a visitor navigates to a webpage where an action map is set to trigger a content offer, the action map doesn't offer the content offer if it is in a terminal state. This feature ensures that a visitor does not receive the same content offer after accepting the offer or indicating that they are not interested in that particular content offer.

Terminal states for content offers are:

- Accepted
- Rejected
- Ignored

For more information, see Define an action map's triggers.

Report metrics and events

The metrics in the Action Map Performance report correlate directly with the event types for web actions. For more information about metrics for content offers, see Monitor a content offer's performance.

Examples: Events methods with content offers

Contents

See examples of how to use Events methods with content offers.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Subscribe to offered events

```
ac('on', 'contentoffer:offered', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to accepted events

```
ac('on', 'contentoffer:accepted', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to rejected events

```
ac('on', 'contentoffer:rejected', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to ignored events

```
ac('on', 'contentoffer:ignored', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to errored events

```
ac('on', 'contentoffer:errored', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to all content offer events

```
ac('on', 'contentoffer:all', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Unsubscribe from content offers in the offered state

```
ac('off', 'contentoffer:offered', eventHandler); // unsubscribes `eventHandler` from
'contentoffer:offered' events
```

Web chat lifecycle

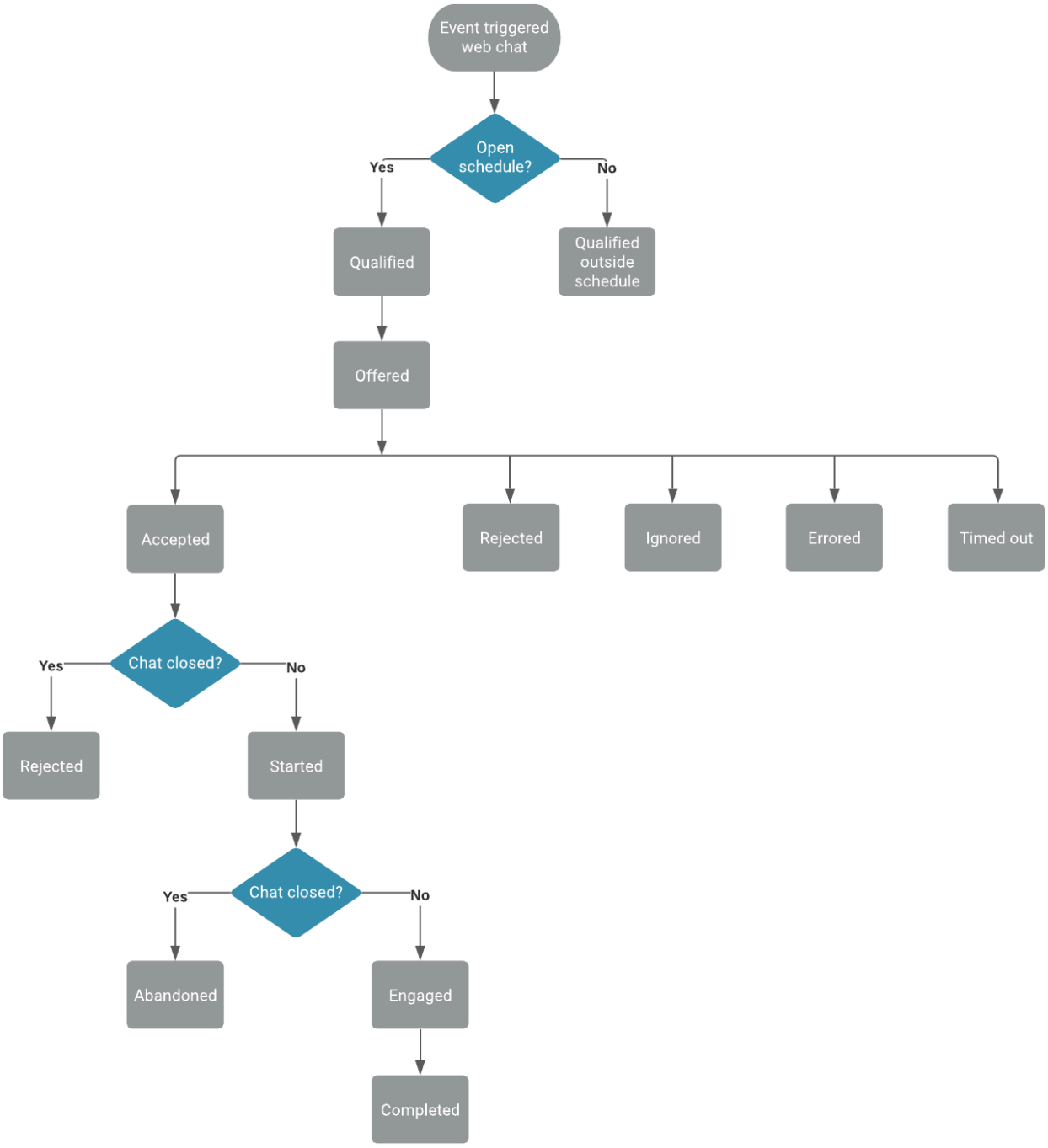
Contents

See the lifecycle of a web chat and the metrics that we capture at each state along the way. Genesys Predictive Engagement uses lifecycle states for reporting and determining the triggering behavior of action maps that use web chats.

Important

This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

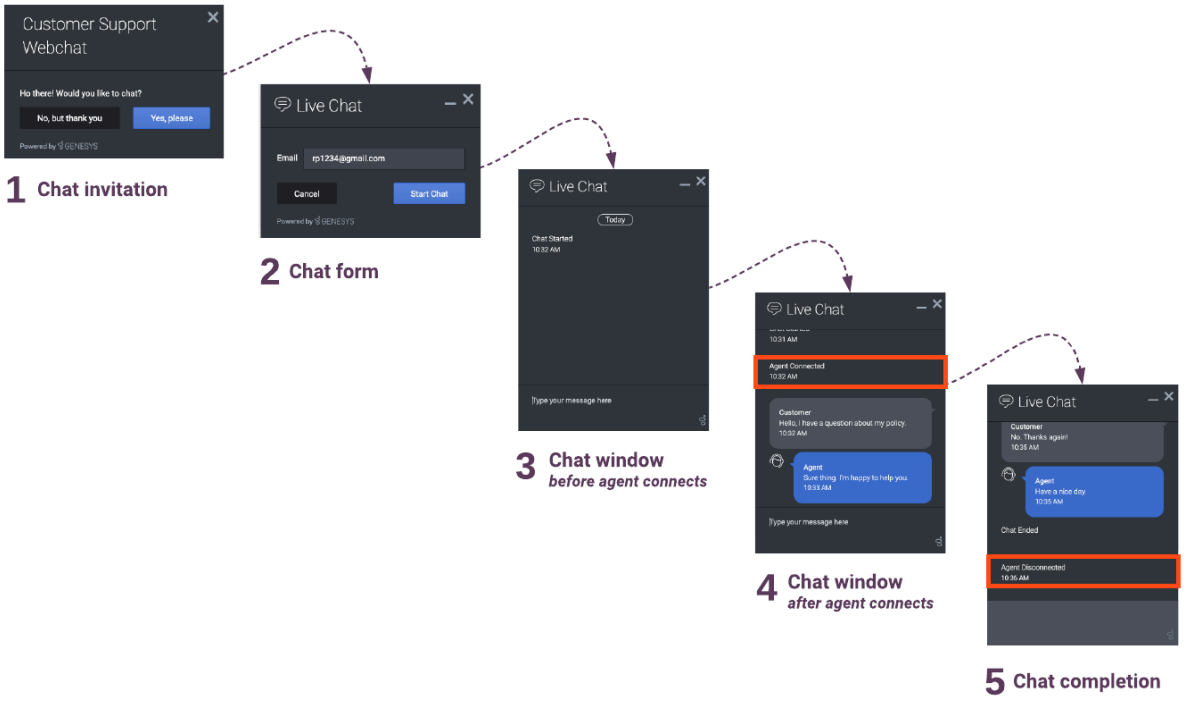
Web chat operations



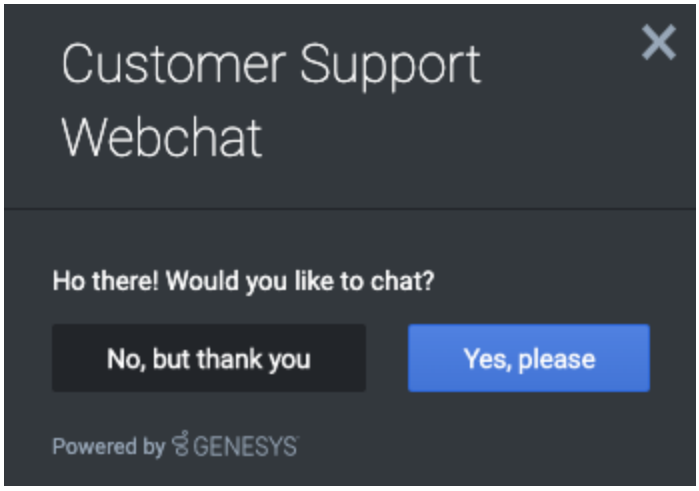
Web chat lifecycle

The following diagram shows the stages that occur during the lifecycle of web chats after offering them to customers. Subsequent sections provide details about specific states, including the events that can occur and the data that is available for use with the Events methods for web actions. The Terminal states section explains how states ensure that customers do not see the same offer to chat repeatedly.

For more information about web chat offering, see Offered action maps.



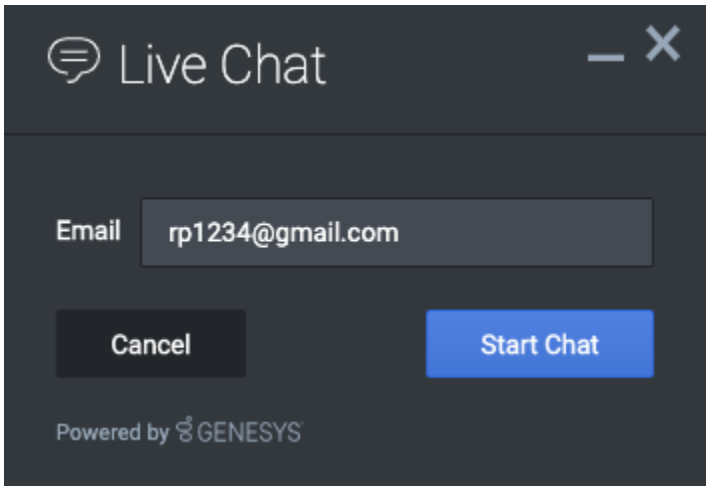
1. Web chat invitation



State	Event	Description	Data collected
offered	Web Actions Offered	Visitor's activity qualified an action map and triggered a chat invitation.	See Event types for web actions.
accepted	Web Actions Accepted	Visitor accepts the chat invitation.	See Event types for web actions.
rejected	Web Actions Rejected	Visitor rejects the chat invitation. This state is a terminal state.	See Event types for web actions.
ignored	Web Actions Ignored	Visitor ignored the invitation by navigating away from or around it. This state is a terminal state. **Note:** This event does not have a corresponding metric in the Action Map Performance Report.	See Event types for web actions.
errored	Web Actions Errored	Error occurred in the widget that prevented the engagement from occurring. **Note:** This event does not have a corresponding metric in the Action Map Performance Report.	See Event types for web actions. Also, the errorMessage field is available.
timed out	Web Actions Timed Out	Chat invitation timed out and was rescinded. This state is a terminal state. **Note:** This event does not have a corresponding metric	See Event types for web actions.

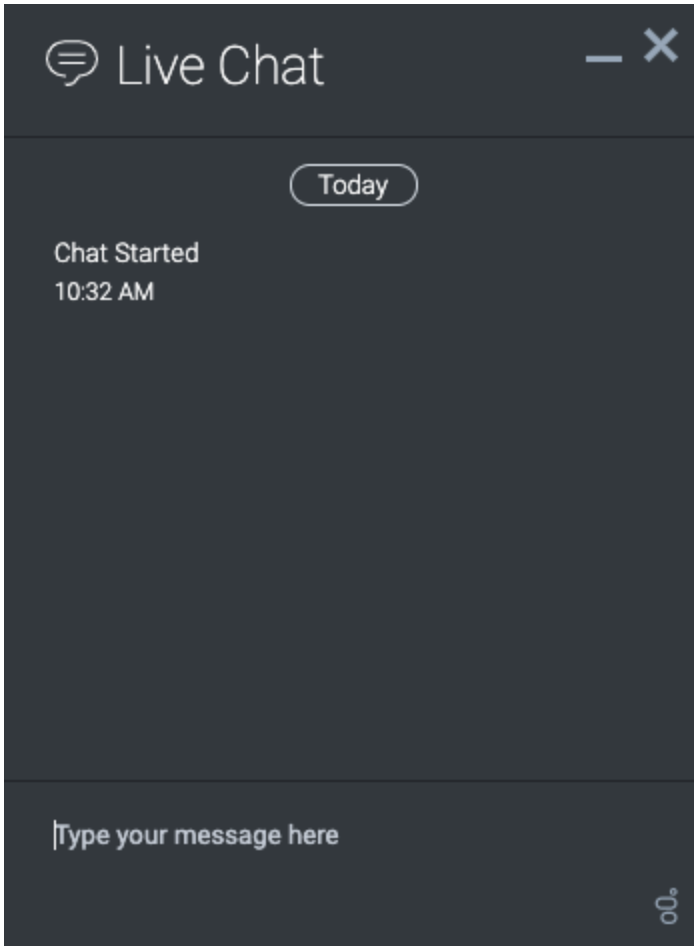
State	Event	Description	Data collected
		in the Action Map Performance Report. The timeout period is configurable through the widget.	

2. Web chat form



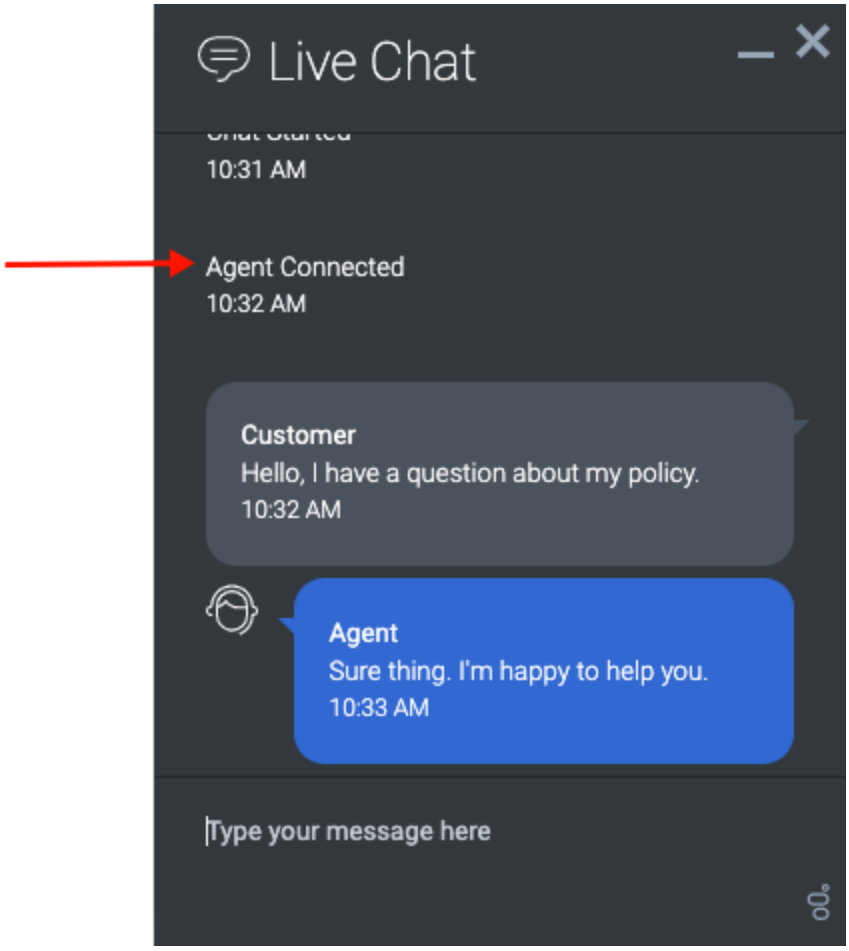
State	Event	Description	Data collected
rejected	Web Actions Rejected	Visitor cancels or closes the form. This state is a terminal state.	See Event types for web actions.

3. Web chat window: before agent connects



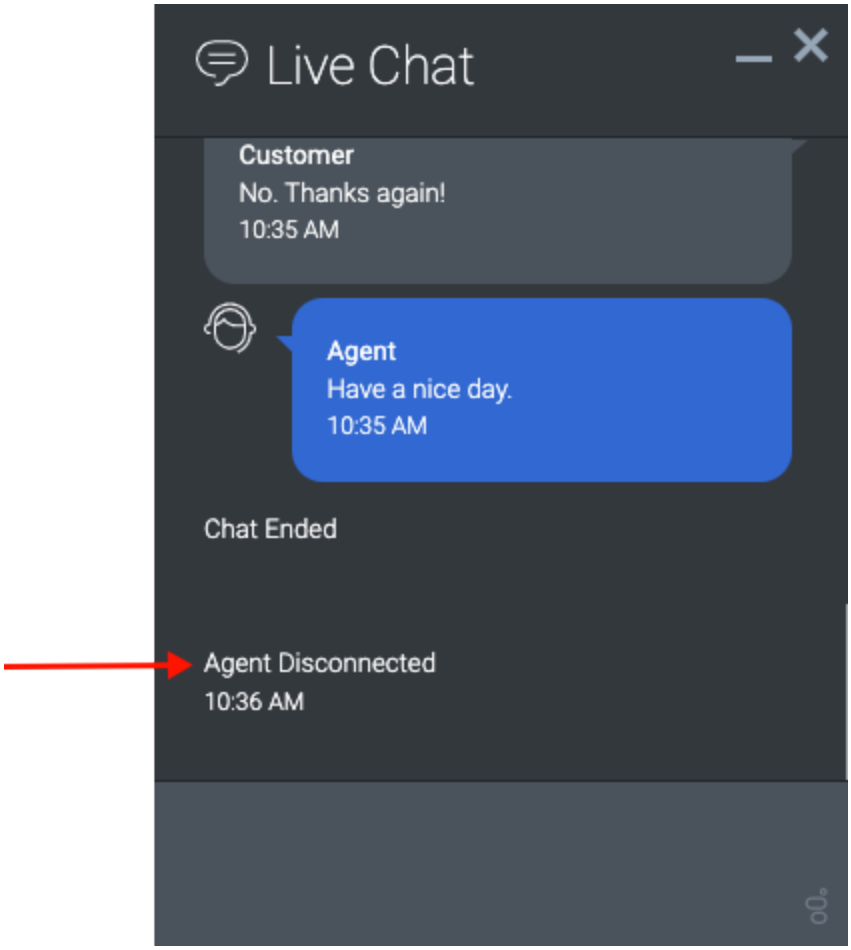
State	Event	Description	Data collected
started	Web Actions Started	After the visitor submits the form, a chat interaction starts.	See Event types for web actions.
abandoned	Web Actions Abandoned	Visitor closes the chat window before an agent connects. This state is a terminal state.	See Event types for web actions.

4 Web chat window: after agent connects



State	Event	Description	Data collected
engaged	Web Actions Engaged	Agent accepts the chat and connects with the visitor. This state is a terminal state.	See Event types for web actions.

5. Web chat completion



State	Event	Description	Data collected
Not applicable/Not tracked	Not applicable/Not tracked	Either the visitor or the agent ends the chat. **Note:** This event does not have a corresponding metric in the Action Map Performance Report.	See Event types for web actions.

Terminal states for web chats

In the web chat lifecycle, certain states are *terminal*, or final, states. If a visitor visits a webpage where an action map is set to trigger a web chat, the action map doesn't offer the web chat if it is in a terminal state. This feature ensures that a visitor does not receive the same offer to chat after accepting the offer already or indicating that they are not interested in that particular chat offer.

Terminal states for web chats are:

- Engaged
- Rejected
- Timed out
- Ignored
- Abandoned

For more information, see Define an action map's triggers.

Report metrics and events

The metrics used in the Action Map Performance report metrics correlate directly with the event types for web actions. For more information about metrics for web chats, see Monitor a web chat's performance.

Examples: Events methods with web chats

Contents

See examples of how to use Events methods with web chats.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Subscribe to offered events

```
ac('on', 'webchat:offered', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to accepted events

```
ac('on', 'webchat:accepted', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to started events

```
ac('on', 'webchat:started', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to engaged events

```
ac('on', 'webchat:engaged', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to rejected events

```
ac('on', 'webchat:rejected', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to ignored events

```
ac('on', 'webchat:ignored', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to errored events

```
ac('on', 'webchat:errored', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to timedout events

```
ac('on', 'webchat:timedout', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Subscribe to all web chat events

```
ac('on', 'webchat:all', (event) => {
  console.log('received %s:%s event', event.actionMediaType, event.actionState, event);
});
```

Unsubscribe from web chats in the offered state

```
ac('off', 'webchat:offered', eventHandler); // unsubscribes `eventHandler` from
'webchat:offered' events
```

Utility methods

Contents

Learn how to simplify calls to the Journey JavaScript SDK collection methods.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Utility methods

Genesys Predictive Engagement utility methods simplify calls to the Journey SDK collection methods.

For more information, see Collect tracking data in Web Tracking API.

- serialize
- dom - ready
- debug

serialize

Contents

Learn how to use the `serialize` method to submit serialized data from a form.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

serialize

The `serialize` method creates a JSON object where the keys are the form elements names, and the value for each key is the value(s) of the corresponding form elements. This method is useful in scenarios such as submitting a form with AJAX, since submit cannot capture the form submission. Instead a record can be invoked passing the serialized form data as its last argument. The serialize method takes the following parameters:

- A unique identifier of the HTML form element
- An (optional) array with the names of the elements to serialize
- A callback function with the serialized form data into a JSON object

```
ac('serialize', 'my-form', function(data) {
  console.log(data['contact-email']);
});
```

dom - ready

Contents

Learn how to specify a function handler to run when the DOM has loaded fully.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

dom - ready

The `ready` method of the `dom` module allows you to specify a function handler to run when the DOM has loaded fully. A webpage cannot be altered safely until the document is ready, Journey SDK starts executing regardless of the DOM readiness, and may even fire events before that. Place any code reacting to these events that need to modify the DOM inside the ready handler.

```
ac('dom', 'ready', function () {
  // safely modify the DOM here
});
```

debug

Contents

Learn how to send helpful messages to the console when you are troubleshooting tracking behavior.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

debug

To turn on debug mode, invoke the debug method. This mode logs helpful messages to the console.

This example shows how to enable debug mode:

```
ac('debug');
```

This example shows how to disable debug mode:

```
ac('debug', false);
```

Web Tracking API

Contents

Learn how to track visitor activity using an API.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

About the Web Tracking API

The Web Tracking API lets you track what visitors do on your website. Tracking data is collected through a series of interactions occurring on your website such as pageviews, button clicks, and custom events. These interactions are grouped into visits and act as a container for the actions that a specific visitor takes on your website.

Visits do not have a predefined duration. Depending on your visitor, the visit may be a few seconds or a couple of hours long. A new visit is created when the visitor has been idle for 30 minutes or more, but they all are linked to the same visitor.

Obtain consent before tracking visitors

> **Important**
> To achieve compliance with GDPR requirements, consider whether you need to obtain a visitor's consent before tracking their data. For more information on using Genesys Predictive Engagement in a GDPR-compliant manner, see GDPR.

To implement tracking after receiving consent, modify the tracking snippet so that the `ac('init')` and `ac('pageview')` are only called when consent is given, as shown in the following example:

```
 (function(a,t,c,l,o,u,d){a['_genesysJourneySdk']=o;a[o]=a[o]||function(){
  (a[o].q=a[o].q||[]).push(arguments)},a[o].l=1*new Date();u=t.createElement(c),

d=t.getElementsByTagName(c)[0];u.async=1;u.src=l;u.charset='utf-8';d.parentNode.insertBefore(u,d)
  })(window, document, 'script', 'https://apps.inindca.com/journey/sdk/js/web/v1/ac.js',
'ac');
if (consentGiven) {
```

```
// Call the ac('init') function to enable tracking
  ac('init', 'a061a3fe-7a80-4b50-9d3b-df88c0f9efad', { region: 'use1' });
  ac('pageview');
}
```

You are responsible for setting the value for the `consentGiven` variable based on the visitor's choice.

Enable web tracking

To enable Web tracking on your website, initialize the Tracking SDK and then call the `pageview` method when a visitor navigates to a new page.

Stop tracking when a visitor revokes consent

If a visitor revokes consent at any point, invoke the `destroy` command to stop tracking and remove all cookies, as shown in the following example.

```
// to disable tracking and delete Genesys Predictive Engagement cookies
ac('destroy');
```

Types of tracked data

Contents

Learn how to track different types of data using Journey SDK.

The Journey JavaScript SDK lets you customize how you collect tracking data for your website. The most basic form of tracking is page view tracking. For page view tracking, Genesys Predictive Engagement records each page a visitor visits. You can also use the Journey JavaScript SDK to record custom visitor activities such as button clicks. For more information see Track custom events.

Track viewed pages

The `pageview` method tracks the webpages that your visitors view. To send a `pageview`, call the `ac` function and pass `pageview` as the first argument.

```
ac('pageview');
```

> **Important**
> By default, the Genesys Predictive Engagement tracking snippet contains the `ac` function.

Track custom events

The `record` method allows you to track custom events, usually as a result of a person interacting with an element or control in your website. For example, the click of a button.

The `record` method takes two parameters:

- The name of the event to record as a string. **Note:** See Guidelines for custom event names.
- An (optional) key-value hash of properties for the event.
- An (optional) callback function that invokes when the request is completed.
- An (optional) callback timeout, in milliseconds, to configure how long to wait when the event takes too long to complete. This option is useful to capture events (such as file downloads) before navigating to a different page/URL, ensuring that the visitor is always redirected.

```
ac('record', 'section_opened');
```

You can also provide extra metadata with your custom event:

```
ac('record', 'button_clicked', {
```

```
  companyName: 'Acme Inc,',
  employees: '100-500'
});
```

The following code sample shows how to record a file download and navigate to the file URL `/files/pricing.pdf` when capturing the event, or after 400 milliseconds:

```
ac('record', 'file.downloaded', {
  name: 'Pricing',
  fileType: 'pdf'
}, function () {
  navigateTo('/files/pricing.pdf');
}, 400);
```

Guidelines for custom event names

> **Important**
>
> Event names:
>
> - Must be less than 255 characters; we recommend keeping them short and descriptive
> - Can contain any combination of alphanumeric characters, underscores, or hyphens only

The suggested format for custom event names is: *`object-delimiter-action`*, where object is the object that was interacted with, and action is the type of interaction that occurred.

Examples of custom event names include:

- `"detected_errors"`
- `"section_failed"`
- `"section_submitted"`
- `"product_added"`
- `"product_removed"`
- `"address_selected"`

customAttributes

Contents

Learn how to use custom attribute to send additional information about your web events.

Custom attributes are optional properties that can be set on a Journey event to pass additional information about the event.

Use customAttribute in the following cases:

- To display the information to an agent handling the interaction with customer or viewing customer's web session.
- To use the information to match segments or to trigger action maps.

Format

To track Journey web events with custom attributes, include the custom attributes in the Journey tracking command arguments.

Ensure that the custom attribute you add to the event following the format below:

- Meaningful name - Since the custom attribute name is displayed as a tooltip for agents watching the customer journey and for administrators who add the event to a segment or an action, ensure you have a meaningful name with a quick recall value. For example: productPrice, totalValue, catalogNumber.
- Associated value - Genesys Cloud supports three primitive JavaScript types and one specific object type:
  - The primitive values supported are number, string, and boolean.
  - The customAttribute object type can have a dataType (of type integer, number, string, and boolean) and a value based on the specified dataType. For example, if the dataType is 'integer', the value could be 123.

In the following example, we have added the custom attribute to the event *product_added* to the *record* command.

```
ac('record', 'product_added', {
          price: 15.99,
          productCode: 'CDE-123',
          productName: 'Product',
          hasBatteries: false,
          quantity: {
                  dataType: 'integer',
                  value: '12'
                       }
});
```

Validation

Ensure that the custom attributes follow the rules below. Invalid custom attributes are dropped and therefore cannot be used by admins nor can be seen by agents.

- The attribute value cannot be **null** or **undefined.**
- The attribute value cannot be an **empty string.**
- The customAttribute must contain both dataType and value. Both the keys must contain a string value, and the value attribute must match the string you specified for the 'dataType'. For example, if the dataType is 'Boolean', the value can be 'True'. The value must not be set to 'Yes'.
- The attribute 'dataType' is associated with the specified event attribute when first processed. To use a different dataType, create an event with a different name or an attribute with a different name.

Event and eventType validation

The following rules apply to event type that contains custom attributes:

- A maximum of 20 attributes is allowed for an event. When the number of attributes is greater than 20, the event is not recorded.
- A maximum of 50 unique attributes is allowed for an event type. When the number of unique attributes is greater than 50, the event types will not store the attribute definitions. However, the event will be processed as normal.
- For changing the event type and all associated custom attribute names and data types, use the Journey Event Types Public API. See Event types.

Form Tracking API

Contents

Learn how to use an API to track form submission and abandonment events, including what information visitors provide in your forms.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

About the Form Tracking API

To track customer activities on your website, you must first deploy the tracking snippet. Then, you can track form-level activity with the Form tracking API.

To track a form, it needs an ID, name, or action field that identifies it uniquely.

By default, the values of the input, select, and text area elements provided in the form (fields with personal information such as name, email, phone number, and company name) are sent with the form submission and abandoned events. The customer profile updates accordingly upon form submission.

> **Important**
> You can use Genesys Predictive Engagement to track visitor activity in a GDPR compliant-manner. However, you need to modify the tracking snippet to be compliant. For more information on how to be compliant with the GDPR requirements, see General Data Protection Regulation (GDPR).

Enable form tracking

To enable form tracking, call `forms:track` and specify the CSS selector of the form(s) that you want to track. For example:

```
ac('forms:track', '.hs-form');
```

To enable tracking for all forms, call the function without specifying the selector. For example:

```
ac('forms:track');
```

To enable tracking for multiple forms based on the CSS selector, specify a group of selectors as a comma-separated list. For example:

```
ac('forms:track', '#firstFormId, #secondFormId');
```

> **Important**
>
> Where the same selector is applied to multiple forms, each form with that selector is tracked separately.


```
ac('forms:track', '#firstFormId, #secondFormId');
```


Example: Track a webpage with 3 forms

Sample webpage with multiple forms

Tracking Forms Example

Get Name Form

First name:

Last name:

Submit

Get Account Information Form

Account No.:

Account branch:

Submit

Register for Newsletter Form

First name:

Last name:

Email address:

Submit

Track all forms in the HTML page

The following code tracks all of the form tags in the HTML page:

```
ac('forms:track');
```

or

```
ac('forms:track', 'form');
```

Track by element IDs

The following code tracks only the Get Account Information form:

```
ac('forms:track', '#get-account-information');
```

The following code tracks both the Get Account Information Form and Get Name Form:

```
ac('forms:track', '#get-account-information, #get-name');
```

Track by element classes

The following code tracks all forms with the `target-class` class. Both the Get Account Information form and Register for Newsletter form are tracked:

```
ac('forms:track', '.target-class');
```

Track by a combination of element IDs and element classes

The following code tracks a form with the `get-name` ID and all forms with the `target-class` class. As a result, all forms shown in the sample are tracked.

```
ac('forms:track', '#get-name, .target-class');
```

Manage form field data

To manage how form field data is sent with form submission and abandonment events, use the `captureFormDataOnAbandon` and `captureFormDataOnSubmit` options with the forms:track method. Set these options to `true` or `false`, depending on the result that you want to obtain.

For example, to record a form abandonment event, but exclude any form field data, set `captureFormDataOnAbandon` to `false`.

```
ac('forms:track', '.hs-form', {
  captureFormDataOnAbandon: false
});
```

> **Important**
>
> If you do not set the `captureFormDataOnAbandon` option, or you set it to anything other than `false`, the option assumes that the value is `true`, and the recorded form abandonment event will contain serialized form data. This data is subject to custom transformation; sensitive fields are excluded. For more information, see Transform data before sending it and Sensitive form fields that are never tracked.

To capture serialized form data for form submission events, set the `captureFormDataOnSubmit` option to `true`.

You can set both options at the same time, as shown in the following example.

```
ac('forms:track', '.hs-form', {
  captureFormDataOnAbandon: false,
  captureFormDataOnSubmit: false
});
```

Transform data before sending it

To configure the format of the data that is sent with the form submission and abandonment events, specify a custom `transform` function in the `forms:track` options. This function receives an object with the form data; it should return an object with the data to send with the form events. You can exclude certain fields, rename fields that are not meaningful, or make any other transformations to the form data before it is sent. The data object is a JSON object of "name-value" pairs, where:

- "name" is the value of the element's ID or name attribute
- "value" is the visitor's input

```
ac('forms:track', 'form', {
  transform: function (data) {
    return {
      deliveryOption: data['radioButton_3'],
      acceptedLicense: data['checkbox_license']
    }
  }
});
```

Sensitive form fields that are never tracked

> **Important**
>
> Passwords, hidden fields, and sensitive fields are never tracked.

To denote sensitive fields, use the following regular expression, after removing non-alphanumeric characters from the field name.

```
/pass|billing|creditcard|cardnum|^cc|ccnum|exp|seccode|securitycode|securitynum|cvc|cvv|ssn|
socialsec|socsec|csc/i.
```

If we track a field that you consider to be sensitive, please contact customercare@genesys.com.

Map traits to link customer records

Contents

Learn how to map multiple records for the same customer to see more complete customer profiles in Live Now.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

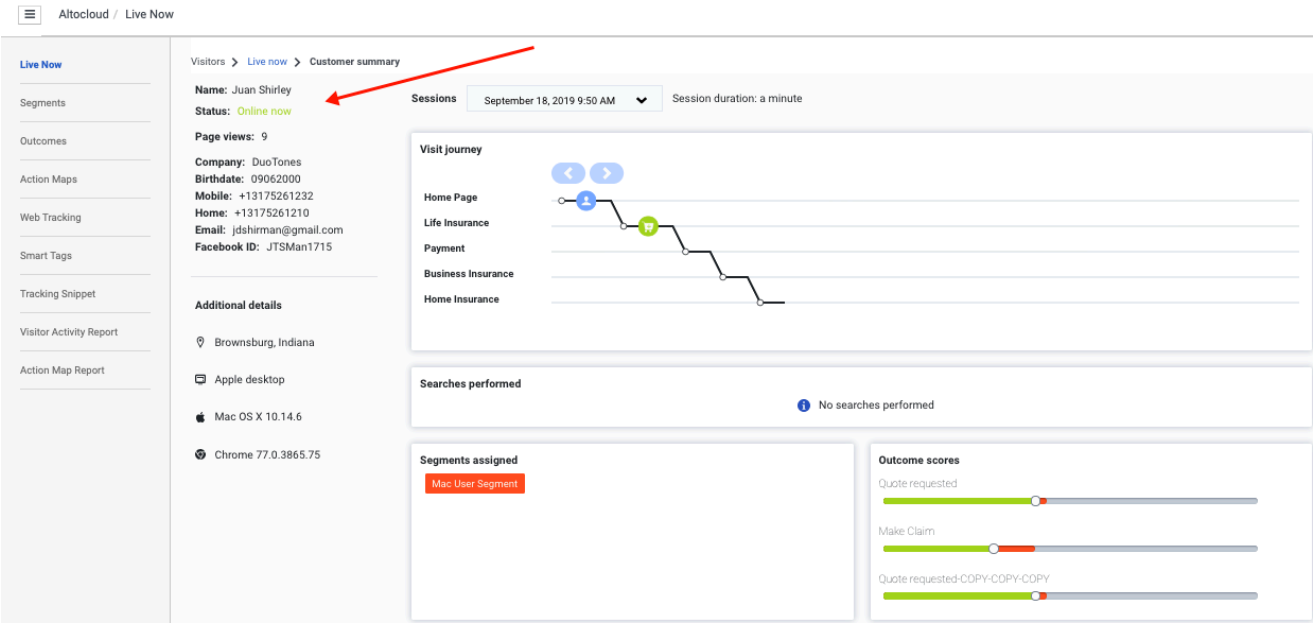
About traits mapping

Traits are properties, such as a customer's email address or phone number. Genesys Predictive Engagement gathers customer traits every time a customer visits a website that you track with the Genesys Predictive Engagement tracking snippet. It's possible to have multiple customer records for the same person. For example, when a customer visits your website multiple times and uses a different browser each time. Because Genesys Predictive Engagement creates a separate record for each instance, the separate customer records may contain only a subset of all the available customer traits. You can map the traits that the separate customer records contain to link the records. Then, you can see the complete customer information in Live Now.

> **Important**
> When the traits mapper links customer records, it preserves the separate customer records. It doesn't consolidate them into a single customer record. Instead, the traits mapper updates all linked customer records with the current traits information.

After linking customer records, the traits mapper updates all the records when new trait information becomes available. It overwrites existing or duplicate traits with the most current trait information.

View mapped traits in the user interface



After you map traits, they appear here:

- Genesys Cloud CX > Admin menu > Live Now > Customer summary (admin view)
- Agent user interface > Journey gadget >
  - Visitor information (Genesys Multicloud CX)
  - Visitor information (PureConnect)

Map traits globally

To start mapping traits, define a global traits mapper when you deploy the Genesys Predictive Engagement tracking snippet on your website. Specifically, when you call init to initialize the Journey JavaScript SDK, identify which attributes to treat as traits. See the following code example.

For more information, see Methods that track events and Mappable traits.

When Genesys Predictive Engagement gathers values for these attributes, they map as traits.

You can also map traits based on specific events.

Example

Map traits for a specific event

You can map specific traits locally instead of globally for specific events. For more information, see

Methods that track events and Mappable traits.

The complete set of map traits for a customer is the union of globally and locally mapped traits. For example, suppose you map the email address field using the global traits mapper, but on one page you ask for the customer's phone number. Both the email address and the phone number map to the customer and both appear in the customer's Live Now profile.

If Genesys Predictive Engagement captures the same data in two places, the most recent trait mapped appears in Live Now. Previous values for mapped traits are not preserved.

Examples of mapped traits

The following examples show how to map attributes as traits. Specifically:

- The attributes, "email" and "emailAddress" map to the trait "email."

Attributes	traitsMapper	Traits

{
"email": "firstname.lastname@somemail.com",
"comment": "This is great",
"section": "support"
}
	[
	{"fieldName": "email"},
	]
		{
		"email": firstname.lastname@somemail.com",
		}

{
"emailAddress": "firstname.lastname@somemail.com",
"comment": "This is great",
"section": "support"
}
	[
	{
	"fieldName": "emailAddress",
	"traitName": "email"
	}
	]
		{
		"email": firstname.lastname@somemail.com",
		}

Methods that track events

Traits mapping can occur whenever there is a tracked event on your website. Specifically, Genesys Predictive Engagement tracks events when you use the following methods:

- init
- pageview
- record
- forms:track

Mappable traits

ID traits

Trait	Example
email	JTS1715@gmail.com

Phone traits

Trait	Example
homePhone	3179871234
cellPhone	3179871235
otherPhone	3179712356
workPhone	8179874321

Name traits

Trait	Example
salutation	Mr.
jobTitle	Manager
givenName	John
middleName	Thomas
familyName	Smith

About modules

Contents

Learn about optional JavaScript files that enhance the functionality that the Journey JavaScript SDK provides.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

About modules

The Journey JavaScript SDK has optional functionality provided as a set of modules. Use the `load` function to load the modules that you need.

Available modules:

- autotrackClick
- autotrackIdle
- autotrackInViewport
- autotrackOfferStateChangesInAdobeAnalytics
- autotrackScrollDepth
- autotrackURLChange

load

Contents

Learn how to add the functionality of a module to the Journey JavaScript SDK.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

Use the `load` function to load modules. When you load a module, its functionality is added to the Journey JavaScript SDK.

Signature

```
ac('load', moduleName, ,[userOptions], [callback])
```

moduleName

- **Description:** name of the module to load. The module must be on the list of valid modules.
- **Type:** string
- **Status:** required

userOptions

- **Description:** configuration for the loaded module
- **Type:** object
- **Status:** module dependent
- **Properties:** module dependent

callback

- **Description:** callback that triggers when the module has finished loading
- **Type:** function
- **Status:** optional (default implementation does nothing)
- **Arguments:**
  - On failure, `load` passes an error (like `InvalidModuleError`)
  - In other cases, `load` passes module-dependent values to the callback

autotrackClick

Contents

Learn how to configure which click events Genesys Predictive Engagement tracks on your websites. This configuration provides more accurate page tracking information for use in segments and outcomes.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `autotrackClick` module tracks when and where a visitor clicks on a webpage. To use `autotrackClick`, configure the click events to track using the following options:

- Config
- Callback

Signature

```
ac('load', 'autotrackClick', config, [callback]);
```

Example

```
ac('load', 'autotrackClick', {
  clickEvents: [
    { selector: 'button.bg-green', eventName: 'green_button_clicked' },
    { selector: 'footer *', eventName: 'footer_clicked' }
  ]
}, function () {
    console.log('"autotrackClick" has been loaded');
});
```

Config (required)

Description: Identifies an array of click events to track.

Type: Object

Properties: See the following table.

Name	Description	Type	Status
selector	String that selects elements. For more information, see https://developer.mozilla.org/en-US/docs/Web/API/Element/matches	String	required
eventName	String used as the event name when an element matching the selector is clicked.	String	required
customAttributes	Field used to send additional information when the selector is clicked. The field can be set with static, predetermined values. Once the autotrackClick module initiates, the value that is available in the value field is applied for the rest of the session. The value cannot be changed dynamically according to user action. There can be more than one field.	Object	optional

Example

To create an event to track the number of clicks on the green button, use the customAttributes as follows:

```
{
  clickEvents: [
    { selector: 'button.bg-green', eventName: 'green_button_clicked', customAttributes:
{clickValue: 100, name: Insurance},
    { selector: 'footer *', eventName: 'footer_clicked' }
  ]
}
```

Use events to track outcome value

Use the attributes within the autotrackClick SDK to track an outcome from action maps. You can

further use the value set within the SDK to the define the value of the outcome. In this example, the value of a click is set at 100. This means that if the value of the outcome stands at 1000, the total number of clicks is 10 with a total value of 1000.

Callback (optional)

When a module loads, callback is executed. No arguments pass to the callback.

autotrackIdle

Contents

Learn how to configure when Genesys Predictive Engagement detects inactivity on a webpage. This configuration provides more accurate page tracking information for use in segments and outcomes.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `autotrackIdle` module tracks when and where a visitor becomes inactive on a webpage. To use `autotrackIdle`, configure the idle events to track using the following options:

- Config
- Callback

Signature

```
ac('load', 'autotrackIdle', [config], [callback]);
```

Example

```
ac('load', 'autotrackIdle'); // This is for all one idle event config with defaults

ac('load', 'autotrackIdle', {
  idleEvents: [
    {}, // This is for all defaults
    { eventName: 'stuck_on_page' },
    { idleAfter: 60 },
    { eventName: 'idle_for_2_min', idleAfter: 120 },
  ]
}, function () {
    console.log('"autotrackIdle" has been loaded');
});
```

Config (optional)

Description: Identifies an array of idle events to track.

Type: Object

Properties: See the following table.

Name	Description	Type	Status
idleAfter	Number of seconds of inactivity after which an event fires. The default is 60 seconds and the minimum is 30 seconds. **Note:** If you specify less than 30 seconds, 30 seconds is used.	Number	optional
eventName	String used as the event name when an element matching the selector is clicked.	String	optional

Example

```
{
  idleEvents: [
    {}, // This is for all defaults
    { eventName: 'stuck_on_page' },
    { idleAfter: 60 },
    { eventName: 'idle_for_2_min', idleAfter: 120 },
  ]
}
```

Callback (optional)

When a module loads, callback is executed. No arguments pass to the callback.

autotrackInViewport

Contents

Learn how to configure which element Genesys Predictive Engagement tracks on your websites as they appear and disappear from the viewport. This configuration provides accurate page tracking information for use in segments and outcomes.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `autotrackInViewport` module tracks when an element becomes visible in the viewport or disappears from the viewport. To use `autotrackInViewport`, configure the elements to track using the following options:

- Config
- Callback

Signature

```
ac('load', 'autotrackInViewport', config, [callback]);
```

Example

```
ac('load', 'autotrackInViewport', {
  inViewportEvents: [
    { selector: 'button.bg-green', eventName: 'green_button_element' },
    { selector: 'footer *', eventName: 'footer_element' }
  ]
}, function () {
    console.log('"autotrackInViewport" has been loaded');
});
```

Config (required)

Description: Identifies an array of elements to track.

Type: Object

Properties: See the following table.

Name	Description	Type	Status
selector	String that selects elements. For more information, see https://developer.mozilla.org/en-US/docs/Web/API/Element/matches	String	required
eventName	String used as the event name when an element matching the selector moves into or outside the viewport.	String	required

Example

```
{
  inViewportEvents: [
    { selector: 'button.bg-green', eventName: 'green_button_element' },
    { selector: 'footer', eventName: 'footer_element' }
  ]
}
```

Callback

When a module loads, callback is executed. No arguments pass to the callback.

autotrackOfferStateChangesInAdobeAnalytics

Contents

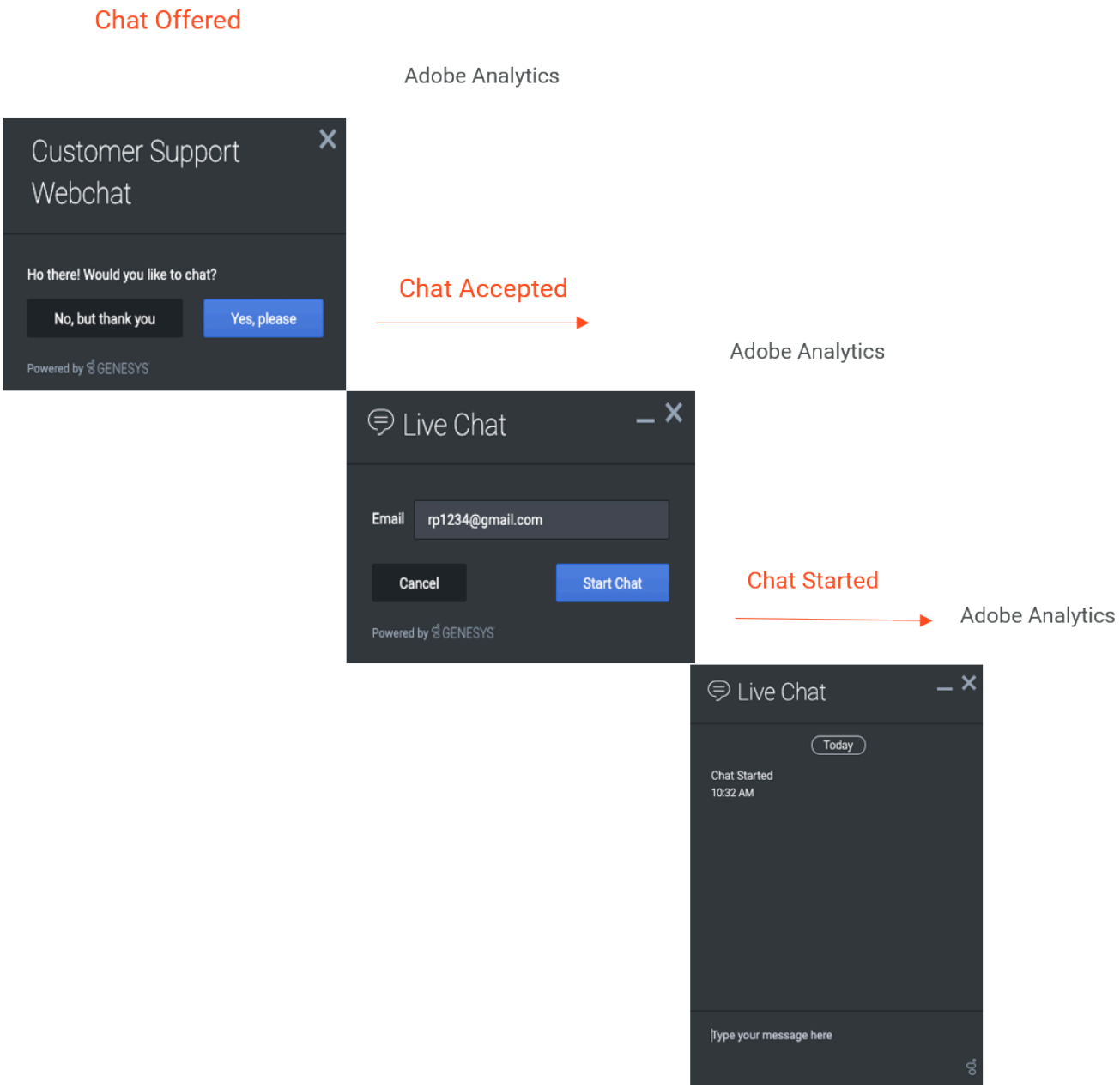
Learn how to track web chat and content offer state changes on a webpage and send them to Adobe Analytics.

Prerequisites

- Required products for integration:
  - Genesys Predictive Engagement with any Genesys platform chat widget
  - Adobe Analytics with latest version of AppMeasurement (to ensure capturing of the Experience Cloud Identifier)
  - Adobe Experience Cloud Identity Service (formerly Marketing Cloud)

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

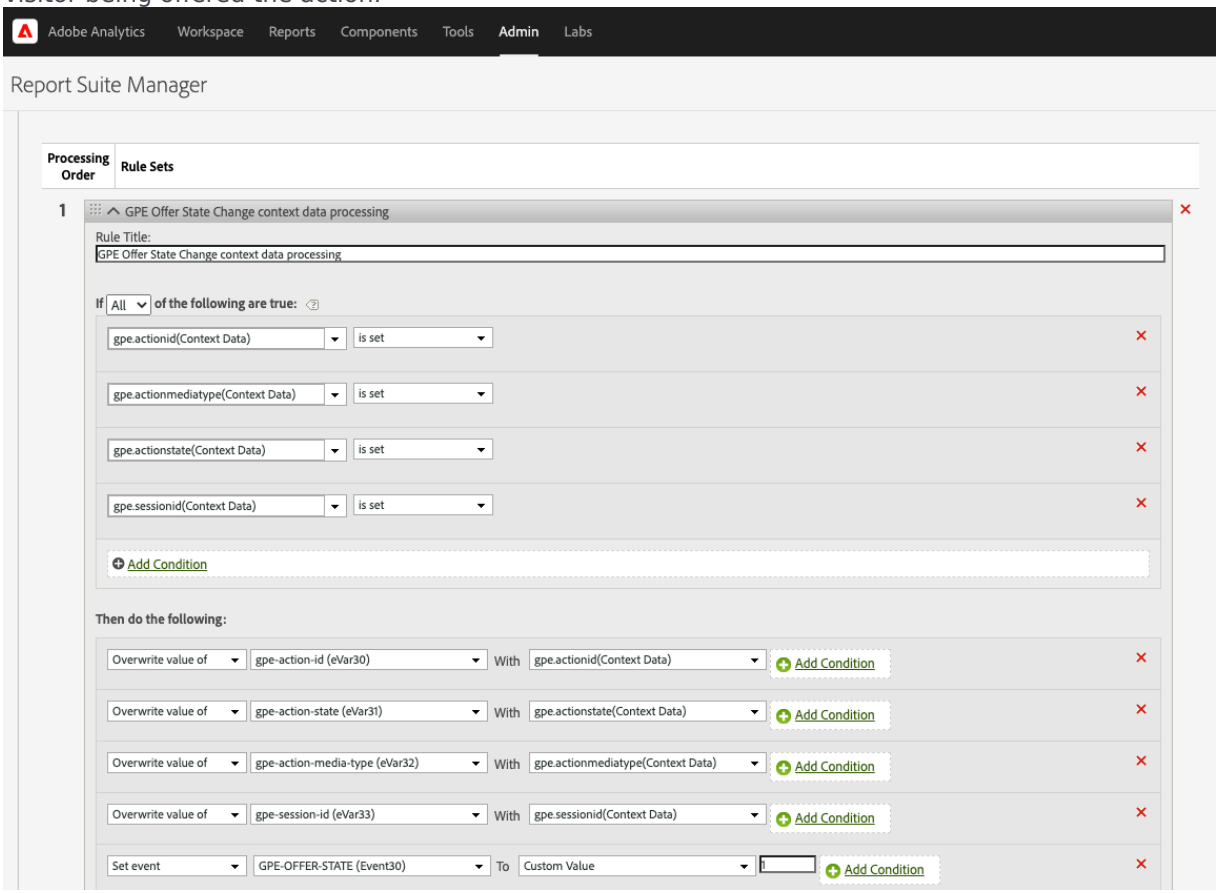


The `autotrackOfferStateChangesInAdobeAnalytics` module tracks web chat and content offer states and sends them in real time to Adobe Analytics for reporting purposes. You can use the data to create reports in Adobe Analytics. This data helps you to evaluate where these events occur in the customer journey and understand how they are influencing your conversion rate.

When the module loads, specify the identifier for the Adobe Analytics library on your webpage.

High-level workflow

1. Enable the following on your website:
   - Genesys Predictive Engagement chat
   - Integration with Adobe Analytics using this `autotrackOfferStateChangesInAdobeAnalytics` module
   - Adobe Analytics tracking including the Experience Cloud Identity Service
2. Genesys Predictive Engagement begins sending data tracking requests for each web chat and content offer state change event in real time to Adobe Analytics.
3. Configure processing rules in Adobe Analytics:
   - Identify which eVars to use and which action states to map. The eVars that you use vary based on the eVars in use in your report suite currently. We recommend that you create a new eVar for each piece of context data that the SDK module sends.
   - Map contextData variables to eVars and, if necessary, map contextData elements to props. The following image is an example of what the processing rules might look like in Adobe Analytics. Once the data is flowing correctly, eVar30 will contain a unique identifier for the proactive chat or content offer. eVar 31 will contain the state associated to the event. eVar32 will identify whether it was a web chat or content offer. eVar33 will contain the web session identifier associated to the visitor being offered the action.



- Increment an event every time the state change tracking call is made.

4. Set up your Adobe Workspace dashboard with custom reporting (see reporting examples).

States tracked and event information sent

States tracked

Following are the available web chat and content offer states tracked for Adobe Analytics.

States	Web Chats	Content Offers
Offered	X	X
Accepted	X	X
Started	X	
Engaged	X	
Rejected	X	X
Ignored	X	X
Errored	X	X
Timedout	X	
Abandoned	X	
Completed	X	

Event information sent

Following are the event information that Genesys Predictive Engagement sends to Adobe Analytics as contextData variables for each state tracked.

Event Information	Web Chats	Content Offers
gpe.actionId	X	X
gpe.actionState	X	X
gpe.actionMediaType	X	X
gpe.sessionId	X	X
gpe.conversationId (available for Genesys Cloud CX customers only)	X	

Reporting examples

Following are some reporting examples that this integration provides:

- Number or percent of Adobe Analytics Reporting segment visitors who accepted a chat and number that proceeded to place an order post-chat
    - Further breakdown by webpage and Adobe segment to reveal patterns of behavior
- Predictive Engagement as an influencing touchpoint on buying behavior
- Quantify orders or sales post-chat with attribution modeling within 30 days of the chat interaction
- Online revenue (transaction value) generated post-chat
- Conversion rate as a percent of overall visitors and as a percent of total chats engaged
- Funnel reporting to show chat impact on conversions
- Page reporting to show the number of pages with the highest number of offered and accepted chats
- Next and previous page flow reporting of web chat engagements
- Break down of web chat engagements by Adobe segments

Signature

```
ac('load', 'autotrackOfferStateChangesInAdobeAnalytics', config, [callback]);
```

Example

```
ac('load', 'autotrackOfferStateChangesInAdobeAnalytics', {
  adobeAnalyticsObjectName: 'customNameForAnalyticsLibrary'
}, function () {
    console.log('"autotrackOfferStateChangesInAdobeAnalytics" has been loaded');
});
```

Config (optional)

Description: Only use this property when the Adobe Analytics tracker library is accessible globally on your webpage but its name is something other than "s." For more information, see Make the Analytics Object Globally Accessible.

Type: Object

Properties: adobeAnalyticsObjectName

Example

The value that you provide is the name of the Adobe Analytics library scoped globally under window. So, if your tracker library is accessible under "window.customNameForAnalyticsLibrary," the config object looks like the

following:

```
{
  adobeAnalyticsObjectName: 'customNameForAnalyticsLibrary'
}
```

Callback (optional)

When a module loads, callback executes. No arguments pass to the callback.

autotrackScrollDepth

Contents

Learn how to configure which scroll milestones Genesys Predictive Engagement tracks on your websites. This configuration provides more accurate page tracking information for use in segments and outcomes.

> **Important**
>
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `autotrackScrollDepth` module tracks when a visitor scrolls to see a specific percentage of a webpage. To use `autotrackScrollDepth`, configure the click events to track using the following options:

- Config
- Callback

Signature

```
ac('load', 'autotrackScrollDepth', config, [callback]);
```

Example

```
ac('load', 'autotrackScrollDepth', {
  scrollDepthEvents: [
    { percentage: 75, eventName: 'scroll_depth_75' },
    { percentage: 100, eventName: 'scroll_depth_100' }
  ]
}, function () {
    console.log('"autotrackScrollDepth" has been loaded');
});
```

Config (required)

Description: Identifies an array of scroll depths to track.

Type: Object

Properties: See the following table.

Name	Description	Type	Status	Default
percentage	String that selects elements. For more information, see https://developer.mozilla.org/en-US/docs/Web/API/Element/matches	Number (0 - 100)	required	NA
eventName	String used as the event name when an element matching the selector is clicked.	String	optional	scroll_depth_ eg if "percentage" = 75, eventName defaults to "scroll_depth_75"

Example

```
{
  scrollDepthEvents: [
    { percentage: 50 },
    { percentage: 100, eventName: 'viewed_full_page' },
  ]
}
```

Callback (optional)

When a module loads, callback is executed. No arguments pass to the callback.

autotrackURLChange

Contents

Learn how to customize tracking for Single Page Application (SPA) websites. Alternatively, use your preferred tag manager to customize and deploy the SPA tracking snippet. For more information, see About event tracking with tag managers. If you are just getting started, read Predictive Engagement tracking snippet.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Description

The `autotrackURLChange` module tracks activity on an SPA webpage when a user clicks through relative links or when software-driven activity changes the URL or browser history.

> **Important**
> The `autotrackURLChange` module is automatically loaded when you load the SPA snippet.

To customize how `autotrackURLChange` tracks user activity, use the following options:

- isUrlChange
- onUrlChange
- callback
- delay

isUrlChange

Description: Checks whether the URL changed since the last check.

Type: Function

Status: Default implementation available; can be overwritten

Returns: Boolean

Arguments: See the following table.

Name	Description	Type	Status
oldUrl	Previous URL before the change.	string	required
newUrl	Possibly changed URL.	string	required

Example

This example shows how to exclude tracking when a visitor clicks relative links on a webpage.

```
ac('load', 'autotrackUrlChange', {
  isUrlChange: function(oldUrl, newUrl) {
    if (oldUrl !== newUrl) {
      if (oldUrl !== '' && newUrl.includes(oldUrl)) {
        return false;
      }
      return true;
    }
    return false;
  }
}, function() { });
```

onUrlChange

Description: Runs when the URL changes through a relative link or SPA routing functionality.

Type: Function

Status: Default implementation available; tracks pageviews through `ac('pageview')`; can be overwritten

Returns: Void

Arguments: See the following table.

Name	Description	Type	Status
newURL	URL that the visitor changed to.	string	required

> Important

> By default, the tracking snippet tracks page views via `ac('pageview')`. If you override the default behavior using `onUriChange`, remember to include this call in the new function manually.

callback

When a module loads, callback is executed. No arguments pass to the callback.

Example

Your SPA page may use routers or relative links to change the page URL without changing the page title. In this case, the default `ac('pageview')` call tracks all page views as occurring on the same page. Visit journey information that appears in Live Now and the agent UI does not reflect actual visitor behavior. To change this behavior, customize the onUrlChange option as shown in the following example.

```
ac('load', 'autotrackUrlChange', {
  onUrlChange: function(newUrl) {
    let customTitle = document.title;
    if (newUrl.includes('marketing')) {
      customTitle = customTitle + 'marketing';
    }
    if (newUrl.includes('contact-us')) {
      customTitle = customTitle + 'contact-us';
    }
    // supply pageOverrides with location and custom title field
    ac('pageview', {
      location: window.location.href,
      title: customTitle
    });
  }
}, function() { });
```

delay

Use this option to set a delay between the time when the SDK realizes there is a URL change and when it sends the `onUrlChange` function.

Example

This example sets a 1-second delay to give the application time to change the page title to match the current URL.

```
ac('load', 'autotrackUrlChange', { delay: 1000 });
```

Exclude URL query parameters

Contents

Exclude irrelevant URL query parameters to improve analytics.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Exclude URL query parameters

Websites often use URL query parameters that are not relevant and don't influence the content displayed to the visitor. For example, in web analytics, parameters such as session IDs and campaign IDs are not informative. We recommend that you use the Journey JavaScript SDK to exclude these query parameters when you register unique page views. Genesys Predictive Engagement doesn't track or include excluded query parameters in analytics reports.

> **Important**
> Be careful when you exclude parameters. For example, if you want to know which products visitors view on your site, do not exclude a query string parameter for a product ID.

To configure Genesys Predictive Engagement to skip certain query parameters:

1. Log in as an administrator.
2. Go to **Global Settings > Tracking Settings**.
3. Provide a comma-separated list of the unwanted parameters in the **Exclude URL Query Parameters** field (for example: sessionid,var1,var2).

Track the #hash portion of the URL fragment

Contents

Track the #hash portion of URL fragments to include relevant information in analytics.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Track the #hash portion of the URL fragment

By default, Genesys Predictive Engagement removes the fragment #hash part of the URL. However, you may want to track the URL fragment to analyze relevant information that it contains.

To include the URL fragments in the URLs tracked:

1. Log in as an administrator.
2. Go to **Global Settings > Tracking Settings**.
3. Enable the **Keep URL fragments** setting.

Before you enable this setting, beware that, pageviews are recorded on page load by default. If your website renders different content based on the URL fragment (without issuing a page reload), you can track these interactions as individual pageviews. To do so, call the `ac('pageview')` method each time the URL fragment changes, passing in the desired page location.

Track a page's canonical URL

Contents

Track the canonical URL of a page for precise pageview counts.

> **Important**
> This article only applies to customers using web chat. If you are a Genesys Cloud CX customer, we encourage you to use the new web messaging feature to replace web chat.

Canonical URLs eliminate duplicate content

Canonical URLs help webmasters and site administrators eliminate duplicate content from analytics reports.

For example,

```
http://www.example.com/blog
```

is treated as a different page than

```
http://www.example.com/blog?sidebar=0
```

even though those URLs display the same content. The side-effect is that one page view to each page is reported instead of two pageviews to a single page.

Canonical URLs in links elements

The canonical URL of a page is often specified in a canonical link element, which can be inserted into a section of a webpage.

For example:

How to track a canonical URL

To use your canonical URLs and override the URL that the ac SDK tracks, configure the *init* call to use the canonical link element value when available.

```
ac('init', 'YOUR-ORGANIZATION-ID', {
  region: 'YOUR-REGION',
  canonicalLink: true
});
```