In this section, you'll find the tracked semantic events that serve as a starting point for AI copilot events. You can extend them based on your own requirements.

This table lists the events that you can track from any conversation:

EVENT	DEFINITION	FIELDS
Conversation Started	When a new conversation begins	`conversationId`
Message Sent	When the first message is added to a thread by user	`conversationId`, `messageId`, `message_body`, `role` (default is `"customer"`)
Message Received	Non-custom response (text/voice) to user prompt by copilot	`conversationId`, `messageId`, `message_body`, `role` (default is `"agent"`)
Conversation Ended	When a conversation is completed	`conversationId`, `message_count`
Action Invoked	When the model or user invokes a capability or tool	`conversationId`, `messageId`, `type`, `action`

EVENT	DEFINITION	FIELDS
Media Generated	When the model generates an image/video/audio	`conversationId`, `messageId`, `type`, `sub_type`
Component Loaded	When a new custom (non-text/voice) component is shown to a user	`conversationId`, `messageId`, `type`
Feedback Submitted	When a user rates a conversation or message	`conversationId`, `messageId`, `rating`
Identify	When a new user is identified anonymously or known	`userId` and/or `anonymousId`
Standard Track Calls	For all events sent to Segment based on user actions taken, like `items purchased`, `support requested`	`conversationId`, `messageId`, `...`

Live chat events

Segment can also track the following live chat events:

- Conversation Started
- Message Sent
- Message Received
- Custom Component Loaded
- Action Invoked
- Media Generated
- Conversation Ended

Event details

This section contains the structure and properties of each AI copilot tracked event.

Conversation Started

The Conversation Started event should be sent when a customer sends their first message.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.

Here's an example of a Conversation Started call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Conversation Started",
  "properties": {
    "conversationId": "1238041hdou"
  }
}
```

Message Sent

The Message Sent event should be sent when a user adds a new message to a thread. The default for `role` is `"customer"`.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.
`messageId`	string	The message's unique identifier.
`message_body`	string	The message's content.
`role`	string	The message's sender; default is `"customer"`.

Here's an example of a Message Sent call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Message Sent",
  "properties": {
    "conversationId": "1238041hdou",
    "messageId": "msg123",
    "message_body": "What's the best stock in the Nasdaq right now?",
  "role": "customer"
  }
}
```

Message Received

The Message Received event should be sent when the copilot gives a non-custom response (either text or voice) to something the user asked.

The default for `role` is `"agent"`. You can extend `role` to different agent type, like `ai_agent`, `human_agent`, `task_automation_agent`, and so on.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.
`messageId`	string	The message's unique identifier.
`message_body`	string	The received message's content.
`role`	string	The message's sender; default is `"agent"`.

```
{
  "userId": "123",
  "action": "track",
  "event": "Message Received",
  "properties": {
    "conversationId": "1238041hdou",
    "messageId": "msg124",
    "message_body": "Thank you for reaching out. How can I assist you today?"
  },
  "role": "agent"
}
```

Conversation Ended

The Conversation Ended event should be sent when a customer or agent explicitly indicates that the conversation has ended or deletes the chat.

This event supports the following semantic property:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.

Here's an example of a Conversation Ended call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Conversation Ended",
  "properties": {
    "conversationId": "1238041hdou"
  }
}
```

Action Invoked

The Action nvoked event should be sent when the copilot or user uses a custom capability or tool, like making a call to an external API.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.
`messageId`	string	The message's unique identifier.
`type`	string	The type of action invoked.
`action`	String	The specific action taken with the tool.
`role`	string	The message's sender; default is `"customer"`.

Here's an example of an Action Invoked call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Action Invoked",
  "properties": {
    "conversationId": "1238041hdou",
    "messageId": "msg125",
    "type": "Inventory Request",
    "action": "check stock level",
  "role": "customer"
  }
}
```

Media Generated

This event should be sent when an image, video, or custom audio is generated by the model.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.
`messageId`	string	The message's unique identifier.
`type`	string	The type of media generated (like `"image"`, `"video"`)

PROPERTY	TYPE	DESCRIPTION
`sub_type`	String	Media data type (like `"gif"`, `"mp4"`, `"wav"`)
`role`	string	The message's sender; default is `"agent"`.

Here's an example of a Media Generated call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Media Generated",
  "properties": {
    "conversationId": "1238041hdou",
    "messageId": "msg126",
    "role": "agent",
    "type": "image",
    "sub_type": "gif"
  }
}
```

Component Loaded

This event should be sent when a new, custom component is shown to the user that isn't text or voice.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.
`messageId`	string	The message's unique identifier.
`type`	string	The type of custom component loaded.

Here's an example of a Component Loaded call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Component Loaded",
  "properties": {
    "conversationId": "1238041hdou",
    "messageId": "msg127",
    "type": "Stock Price Chart"
  }
}
```

Feedback Submitted

This event should be sent when a user rates a conversation or message.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`conversationId`	string	The conversation's unique identifier.
`messageId`	string	The message's unique identifier.
`rating`	number	The rating given by the user.

Here's an example of a Feedback Submitted call:

```
{
  "userId": "123",
  "action": "track",
  "event": "Feedback Submitted",
  "properties": {
    "conversationId": "1238041hdou",
    "messageId": "msg128",
    "rating": 5
  }
}
```

Identify

This event should be sent when a new user is identified, either anonymously or as a known user.

This event supports the following semantic properties:

PROPERTY	TYPE	DESCRIPTION
`userId`	string	The user's unique identifier.
`anonymousId`	string	The user's anonymous identifier (if applicable).

Here's an example of an Identify call:

```
{
  "userId": "123" || "anonymousId" : "768923ihuy32",
  "action": "identify",
  "properties":
}
```

Standard Track calls

When sending events to Segment based on user actions, like items purchased or support requested, make sure to include relevant identifiers for accurate tracking and analysis.

These identifiers include `conversationId` and `messageId`, among others, depending on the specific tracked action:

IDENTIFIER	DESCRIPTION
`conversationId`	The conversation's unique identifier. This identifier is crucial to tracking actions within a messaging or support context.
`messageId`	The message's unique identifier. This identifier is especially important for actions like messages read, media generated, or feedback submitted.

For example, to track an event where a user makes a purchase, the standard Track call could look like this:

```
{
  "userId": "user123",
  "action": "track",
  "event": "Item Purchased",
  "properties": {
    "conversationId": "conv456",
    "messageId": "msg789",
    "itemId": "item101112",
    "itemName": "Super Widget",
    "itemPrice": 19.99,
    "currency": "USD"
  }
}
```

This page was last modified: 04 Apr 2024

Need support?

Questions? Problems? Need more info? Contact Segment Support for assistance!

Visit our Support page

Help improve these docs!

Edit this page

Request docs change

Was this page helpful?

Yes

No

Get started with Segment

Segment is the easiest way to integrate your websites & mobile apps data to over 300 analytics and growth tools.

Your work e-mail

Request Demo

or

Create free account

Privacy

Terms

Website Data Collection Preferences