



SAS® Customer Intelligence 360: Programming Interfaces

About the APIs	2
Collecting Data with the Events APIs	2
About the Events APIs	2
Collecting Data with the JavaScript Events API	3
Server-Side Events API	67
Collecting Mobile Event Data	116
About Mobile Events	116
Add to Cart Events	116
Cart Events	118
Internal Search Events	123
Product View Events	123
Promotion Events	125
Content Request API	126
About the Content Request API	127
Prerequisites for Using the Content Request API	129
Send a Request to the Content Request API	129
Retrieve Spot Information	138
REST APIs	139
Container Image Management API	140
Copy Item API	146
Decisioning Subject Contact API	151
Marketing Decisioning API	155
Digital Assets API	159
Marketing Administration API	178
Marketing Audience API	185
Marketing Data API	194
Marketing Execution API	208
Marketing Design API	217
Marketing Gateway API	223

Plan API	224
System for Cross-Domain Identity Management (SCIM) API	230
Workflow API	235
Developer Resources By API Collection	246

About the APIs

SAS Customer Intelligence 360 provides multiple ways to interact with the system and your site, such as JavaScript APIs, mobile SDKs, and REST APIs.

In addition to the developer documentation for these APIs, you can refer to the following information in the user documentation:

- [SAS Customer Intelligence 360: Developer's Guide for Mobile Applications](#)
- Managing customer data:
 - ["Import Data through the REST API" in SAS Customer Intelligence 360: Administration Guide](#)
 - ["Exporting Data" in SAS Customer Intelligence 360: Administration Guide](#)
 - ["GDPR Delete of Identity Data" in SAS Customer Intelligence 360: Administration Guide](#)
- Integrating with other systems through the external API gateway:
 - ["Streaming from SAS Customer Intelligence 360" in SAS Customer Intelligence 360: Administration Guide](#)
 - ["Integrating with Content Management Systems" in SAS Customer Intelligence 360: Administration Guide](#)

Collecting Data with the Events APIs

About the Events APIs

To collect data from users that are using your web site, you can call functions in the JavaScript Events API to record their interactions. When you use this API, you can have complete control over how events are collected from users. For more information, see ["Using the JavaScript Events API, Version 2"](#).

You can use the JavaScript Events API to collect data from mobile users by integrating your mobile code with the API's features. For more information, see ["About Mobile Events"](#).

Collecting Data with the JavaScript Events API

Using the JavaScript Events API, Version 2

About the JavaScript Events API

The JavaScript Events API provides a minimal footprint and enables you to perform these tasks:

- [send events](#) to SAS 360 Discover
- [request definitions of spots and creatives](#)

This API gives you full control over which events to send and when to send them. This level of control is in contrast to the default configuration of the SAS tag, which sends a default set of events automatically. The JavaScript Events API is particularly useful when you are working with single-page applications (created with React, Angular, or similar frameworks).

The JavaScript Event API is part of the [SAS tag](#), so it is available on any page that is configured to work with SAS Customer Intelligence 360.

Note: SAS support can assist with technical issues for the SAS tag and the JavaScript Events API, but they cannot provide guidance on website design. Support is limited to the functionality of the SAS tag and the related features.

In addition, the SAS tag and related APIs are limited by a customer's cookie and tracking settings. For more information, see [“Limitations on Data Collection with Web Events” in SAS Customer Intelligence 360: User's Guide](#).

The SAS tag defines a global object named `ci360`, and this object has functions that enable you to send API calls. The `ci360` object is initialized automatically when the SAS tag loads on your page. You can send events through the API as soon as the snippet for the SAS tag is run. You do not need to wait for the API to return a ready state.

TIP If you use the JavaScript Events API with mobile or hybrid applications, you can set the default domain when you insert the tag snippet. For more information, see [“Override the Domain When a Page Loads” in SAS Customer Intelligence 360: Administration Guide](#).

Send Events

How to Send an Event

To send an event, use the `ci360("send", <payload>)`, where *payload* is the set of parameters for the event type. Here is additional information about parameters:

4

- Each event type has type-specific parameters.
- You can add custom parameters if you need to.
- Parameters can be set as name/value pairs or as name/object pairs. For example, this parameter is a name/value pair:

```
"param1": "my_value1"
```

This parameter is set as a name/object pair:

```
"param2": {  
  "value": "my_value2"  
}
```

Note: Be aware of these limitations and guidelines for collecting data with the JavaScript Events API:

- The number of events that is collected for each session is limited to a maximum of 2,000.
- For any attributes that are collected, the value is truncated to 1,000 characters. This applies to form field values, cookie values, JavaScript variables, page elements, and so on.
- Event names must be unique across all event types, including across subfolders. In addition, events cannot have the same name as rules in global settings (such as [Traffic Sources](#) or [Page URLs](#)).
- Currency values should be sent as a numeric type with a period for the decimal separator and no thousands separator. Currency values that are collected through API events are not affected by the Formats Settings page in the user interface.

When you submit an event, the JavaScript Events API returns a Boolean value `true` to the console (or standard output stream) when the event is sent successfully. The value `false` is returned if the event cannot be sent.

Obscuring Values in the JavaScript Events API

The value for every parameter can be obscured. There are two ways to obscure a value, depending on where the parameter is set:

- To obscure values for top-level parameters, include an `obfuscateFields` parameter that lists the parameters to obscure in a string representation of an array. For example:

```
ci360("send", {  
  "eventName": "<event_type>",  
  "param1": "my_value1",  
  "param2": "my_value2",  
  "obfuscateFields": "["param1", "param2"]"  
});
```

- To obscure field values in a [change event](#) or a [submit event](#), include an `obscure` parameter on the exact parameter that you want to obscure. Set the value to `true`.

For example:

```
ci360("send", {  
  "eventName": "submit",  
  "location": "theLocation",  
  "referrer": "theReferrer",  
  "id": "theId",  
  "obscure": true  
});
```

```

"name": "theName",
"fields": [
  {
    "name": "obscuredName",
    "value": "obscuredValue",
    "type": "text",
    "alt": "altVal",
    "changes": "1",
    "id": "obscuredId",
    "obscure": "true"
  },
  {
    "name": "visibleName",
    "value": "visibleName",
    "type": "text",
    "alt": "altVal",
    "changes": "1",
    "id": "visibleId"
  }
]
});

```

Using the JavaScript Events API with Tasks

To use events that are sent with the JavaScript Events API, there must be a corresponding JavaScript API event that is defined in the user interface. This event contains an API event key, which you then send as an attribute (`apiEventKey`) in the JavaScript event itself.

The event in the user interface acts as a listener for any JavaScript events that contain that key, and it enables you to use JavaScript events in the same way you use other event types (for task targeting, data views, and so on).

Note: For some event types, the JavaScript Events API calculates and sends additional properties automatically. Because you cannot customize their values, these properties are not listed in the tables for each event. However, you can still use the additional properties in the user interface (for features such as targeting).

Complete these steps:

- 1 In SAS Customer Intelligence 360, create an event with a JavaScript API subtype. On the Event Identifier page, specify a value for **API Event Key**. For more information, see [“Creating JavaScript API Events for Web” in SAS Customer Intelligence 360: User's Guide](#).
- 2 When you send an event from the JavaScript Events API, include the `apiEventKey` attribute. Set the value to the same value that you used for **API Event Key** in the user interface.

For example, this is how you set the `apiEventKey` attribute in a Product View event:

```

ci360("send", {
  "eventName": "productView",
  "apiEventKey": "myProdViewKey",
  "id": "theId",
  "name": "theName",
  "sku": "theSku",
  "quantity": "theQuantity",
  "unitPrice": "theUnitPrice",
  "group": "theGroup",

```

```

    "availabilityMessage": "theAvailabilityMessage",
    "shippingMessage": "theShippingMessage",
    "savingsMessage": "theSavingsMessage"
  });

```

Use an API event key with these events in the JavaScript Events API:

- “Cart Events”
- “Cart-Action Events”
- “Custom Events”
- “Internal-Search Events”
- “Product View Events”
- “Promotion Events”
- “Visibility-Change Events”

Event Types

Cart Events

Cart events contain general information about a cart, such as the total value, tax, shipping cost, payment type, delivery type, and other shipping and billing information. Cart events can also contain cart item objects, which include information about individual items in a user’s cart.

Cart events populate the [ORDER_SUMMARY](#) table. Cart items that are embedded in the event populate the [ORDER_DETAILS](#) table.

Here are the parameters for this event type:

Table 1 Parameters for Cart Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
action	Required	String	ORDER_SUMMARY.RECORD_TYPE	The action that is associated with this cart. You must use one of these values: ■ cartview ■ checkout ■ purchase
eventName	Required	String	For internal use only	The event. The value must be cart .
apiEventKey	Required	String	For internal use only	The unique key that is associated with an event defined in the user interface. For

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				more information, see “Using the JavaScript Events API with Tasks”. Depending on the value of the action attribute, this event can be associated with one of these events in the user interface: ■ Cart View event ■ Checkout View event ■ Purchase View event
id	Required	Number, String	ORDER_SUMMARY.CART_ID	The customer’s unique cart ID. The maximum length for this value is 42 characters. When a customer creates a cart, completes a purchase, or cancels a cart, the same cart ID is associated with all of these actions. The ID can be used externally to associate this cart with your external commerce system. Ensure that ID values are unique for each customer’s cart. JavaScript API calls bypass the internal verification to ensure unique IDs for carts, so give each cart a unique ID to avoid duplicate data records.
name	Required	String	For informational purposes only	The name of the cart.
billingCity	Optional	String	ORDER_SUMMARY.BILLING_CITY_NAME	The city where the invoice should be sent. The maximum length for this value is 390 characters.
billingCountry	Optional	String	ORDER_SUMMARY.BILLING_COUNTRY_NM	The country where the invoice should be sent. The maximum length for this value is 85 characters.
billingPostalCode	Optional	String	ORDER_SUMMARY.BILLING_POSTAL_CD	The billing code where the invoice should be sent. The maximum length for this value is 10 characters. For example, in the United States this value could be the ZIP code.
billingRegion	Optional	String	ORDER_SUMMARY.BILLING_STATE_REGION_CD	The region where the invoice should be sent. The maximum length for this value is 256 characters.

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				For example, in the United States this value could be the state.
deliveryType	Optional	String	ORDER_SUMMARY.DELIVERY_TYPE_DESC	The delivery type. The maximum length for this value is 42 characters. For example, the value might be USPS or FEDEX .
items	Optional	Object	Members of this object are written to the ORDER_DETAILS table.	An object that defines one or more cart items. See Table 2 for the parameters that define cart items. Cart events can be sent with no cart items.
orderId	Optional	Number, String	ORDER_SUMMARY.ORDER_ID	The customer's unique order ID. When a customer completes an order, the order ID is the unique identifier. Ensure that order IDs are unique for each customer's order. JavaScript API calls bypass the internal verification to ensure unique IDs for orders, so give each order a unique ID to avoid duplicate data records.
paymentType	Optional	String	ORDER_SUMMARY.PAYMENT_TYPE_DESC	The payment type. The maximum length for this value is 42 characters. For example, the value might be VISA or CHECK .
shippingCity	Optional	String	ORDER_SUMMARY.SHIPPING_CITY_NM	The city where the cart is being shipped. The maximum length for this value is 390 characters.
shippingCost	Optional	Number, String	ORDER_SUMMARY.SHIPPING_AMT	The cost of shipping. The value must be an integer or decimal.
shippingCountry	Optional	String	ORDER_SUMMARY.SHIPPING_COUNTRY_NM	The country where the cart is being shipped. The maximum length for this value is 85 characters.
shippingPostcode	Optional	String	ORDER_SUMMARY.SHIPPING_POSTAL_CD	The shipping code where the cart is being shipped. The maximum length for this value is 10 characters. For example, in the United States this value could be the ZIP code.

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
shippingRegion	Optional	String	ORDER_SUMMARY.SHIPPING_STATE_REGION_CD	The region where the cart is being shipped. The maximum length for this value is 256 characters. For example, in the United States this value could be the state.
tax	Optional	Number, String	ORDER_SUMMARY.TOTAL_TAX_AMOUNT	The tax amount. The value must be an integer or a decimal.
totalCartValue	Optional	Number, String	ORDER_SUMMARY.TOTAL_PRICE_AMOUNT	The total value for goods in the cart. The value must be an integer or a decimal.

Cart item objects are defined with the following parameters:

Note: Invalid values for cart items (such as values that exceed that maximum character limit) do not return an error from the API. Cart items that contain invalid values are dropped, and valid cart items are retained.

Table 2 Parameters for Cart Items

Parameter	Required or Optional	Data Type	Table Location (as Table.Column)	Description
availabilityMessage	Optional	String	ORDER_DETAILS.AVAILABILITY_MESSAGE_TEXT	A message related to item availability. The maximum length for this value is 650 characters. For example, the value could be back-ordered .
group	Optional	String	ORDER_DETAILS.PRODUCT_GROUP_NM	A group that the cart item belongs to. The maximum length for this value is 130 characters. For example, the value could be sporting goods .
id	Required	Number, String	ORDER_DETAILS.PRODUCT_ID	The ID of the cart item. The maximum length for this value is 130 characters. Note: You must define at least one of these parameters: ■ id ■ name

Parameter	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				■ sku
name	Required	String	ORDER_DETAILS.PRODUCT_NM	The name of the cart item. The maximum length for this value is 130 characters. Note: You must define at least one of these parameters: ■ id ■ name ■ sku
sku	Required	Number, String	ORDER_DETAILS.PRODUCT_SKU	The SKU value for the cart item. The maximum length for this value is 100 characters. Note: You must define at least one of these parameters: ■ id ■ name ■ sku
quantity	Optional	Number, String	ORDER_DETAILS.QUANTITY_AMT	The number of this item that exists in the cart. The value must be an integer or decimal.
savingsMessage	Optional	String	ORDER_DETAILS.SAVINGS_MESSAGE_TEXT	A message related to discounts. The maximum length for this value is 650 characters.
shippingMessage	Optional	String	ORDER_DETAILS.SHIPPING_MESSAGE_TEXT	A message related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.
unitPrice	Optional	Number, String	ORDER_DETAILS.UNIT_PRICE_AMT	The price for a single unit of the cart item. This value must be an integer or a decimal.

This is an example of a cart event with two cart items:

```
ci360("send", {
  "eventName": "cart",
  "action": "cartview",
  "apiEventKey": "myCartApiKey",
  "id": "12345",
  "orderId": "theOrderId",
  "name": "theName",
  "billingCity": "billCity",
```

```

"billingCountry": "billCountry",
"billingPostcode": "27517",
"billingRegion": "billRegion",
"deliveryType": "delType",
"paymentType": "payType",
"shippingCity": "shipCity",
"shippingCost": "2.00",
"shippingCountry": "shipCountry",
"shippingPostcode": "27517",
"shippingRegion": "shipRegion",
"tax": "1.23",
"totalCartValue": "4.56",
"items": [
  {
    "id": "itemId1",
    "name": "itemName1",
    "sku": "itemSku1",
    "quantity": "3",
    "unitPrice": "1.23",
    "group": "theGroup",
    "availabilityMessage": "availMessage",
    "shippingMessage": "shipMessage",
    "savingsMessage": "savingsMessage"
  },
  {
    "id": "itemId2",
    "name": "itemName2",
    "sku": "itemSku2",
    "quantity": "2",
    "unitPrice": "4.56",
    "group": "theOtherGroup",
    "availabilityMessage": "availMessage2",
    "shippingMessage": "shipMessage2",
    "savingsMessage": "savingsMessage2"
  }
]
});

```

Cart-Action Events

Cart-action events track actions related to adding, removing, and updating items in a cart. Each cart-action event corresponds to one item in the cart.

Cart-action events populate the [CART_ACTIVITY_DETAILS](#) table.

Note: When you create a cart-action event, set parameter values that are consistent with your external commerce system. It is important to coordinate these parameters with other systems so that analysis and reports are consistent with what you expect.

Here are the parameters for this event type:

Table 3 Parameters for Cart-Action Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
action	Required	String	CART_ACTIVITY_DETAILS.ACTIVITY_CD	The cart action that is performed. You must use one of these values: ■ add ■ update ■ remove Note: When you specify the remove action, the <code>cart_type</code> attribute is required.
eventName	Required	String	For internal use only	The event. The value must be cartaction .
apiEventKey	Required	String	For internal use only	The unique key that is associated with an event defined in the user interface. For more information, see “Using the JavaScript Events API with Tasks”. This event can be associated with an Add to Cart event in the user interface.
cartId	Required	Number, String	CART_ACTIVITY_DETAILS.CART_ID	The cart ID. When a customer creates a cart, completes a purchase, or cancels a cart, the same cart ID is associated with all of these actions. The ID can be used externally to associate this cart with your external commerce system. Ensure that ID values are unique for each customer’s cart. JavaScript API calls bypass the internal verification to ensure unique IDs for carts, so give each cart a unique ID to avoid duplicate data records. Note: You must specify at least one of the following parameters: ■ <code>cartId</code> ■ <code>cartName</code>
cartName	Required	String	For informational purposes only	The cart name. Note: You must specify at least one of the following parameters: ■ <code>cartId</code>

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				■ cartName
cartType	Optional	String	For informational purposes only	The cart type. Note: This parameter is required when the <code>action</code> parameter is set to <code>remove</code>. Use one of these values: ■ cartview ■ checkout
productId	Required	Number, String	CART_ACTIVITY_DETAILS.PRODUCT_ID	The ID for the product. The maximum length for this value is 130 characters. Note: You must specify at least one of the following parameters: ■ productId ■ productName ■ productSku ■ id ■ name ■ sku
productName	Required	String	CART_ACTIVITY_DETAILS.PRODUCT_NM	The name of the product. The maximum length for this value is 130 characters. Note: You must specify at least one of the following parameters: ■ productId ■ productName ■ productSku ■ id ■ name ■ sku
productSku	Required	Number, String	CART_ACTIVITY_DETAILS.PRODUCT_SKU	The SKU value for the product. The maximum length for this value is 100 characters. Note: You must specify at least one of the following parameters: ■ productId ■ productName ■ productSku ■ id

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				■ name ■ sku
cartType	Optional	String	For internal use only	The value must be cartview or checkout. Note: When the action parameter is set to remove, the cartType parameter is required.
group	Optional	String	CART_ACTIVITY_DETAILS.PRODUCT_GROUP_NM	A group that the cart item belongs to. The maximum length for this value is 130 characters. For example, the value could be sporting goods.
quantity	Optional	Number	CART_ACTIVITY_DETAILS.QUANTITY_VAL	The number of this item that exists in the cart. The value must be an integer or decimal.
unitPrice	Optional	Number	CART_ACTIVITY_DETAILS.UNIT_PRICE_AMT	The price for a single unit of the cart item. This value must be an integer or a decimal.

This is an example of a cart-action event:

```
ci360("send", {
  "eventName": "cartaction",
  "action": "add",
  "cartId": "12345",
  "apiEventKey": "myCartActionKey",
  "cartName": "name",
  "productId": "prodId",
  "productName": "prodName",
  "productSku": "prodSku"
});
```

Change Events

Change events track changes to a field and the number of attempts that has been made to submit its value (through a form or button press).

Change events populate the [FORM_DETAILS](#) table. They can be used to trigger parts of UI-based events such as steps in a [Business Process event](#).

TIP Unless you need to send a raw event of this type, consider sending a [custom JavaScript event](#) instead. Custom events enable more flexibility because you send any event data as custom attributes. You can then use this data with the corresponding custom JavaScript API event in the user interface.

Here are the parameters for this event type:

Table 4 Parameters for Change Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be change .
field	Required	Object	The table fields that are populated depend on the item that is sent: ■ name is sent to FORM_DETAILS.FORM_FIELD_NM. ■ value is sent to FORM_DETAILS.FORM_FIELD_VALUE. ■ changes is sent to FORM_DETAILS.CHANGE_INDEX_NO.	An object that defines the field that was changed. This object should contain only one field. A field is defined like this example: <pre>"field": [{ "name": "field name", "value": "thisValue", "changes": 1 }]</pre> The changes parameter is an integer that specifies how many times a field is modified. Note: A field must contain the following parameters: name, value, and changes. To obscure values, add the obscure parameter and set a string value to true. For example: <pre>"field": [{ "name": "obscuredFieldName", "value": "obscuredValue", "changes": 1, "obscure": "true" }]</pre>
id	Required	String	FORM_DETAILS.FORM_NM	An ID that is associated with the change event, field, or parent form.
attempts	Optional	Number	FORM_DETAILS.ATTEMPT_INDEX_CNT	The number of attempts that has been made to submit the field value. This value must be an integer. If a value is not provided, the default is set to 1.

This is an example of a change event with a custom name parameter and field parameter:

```
ci360("send", {
  "eventName": "change",
  "attempts": 1,
  "id": "theId",
  "name": "theName",
  "field": {
    "value": "field1Val",
    "type": "text",
    "name": "field1Name",
    "alt": "altVal",
    "changes": 2,
    "id": "field1Id"
  }
});
```

This is an example of a change event that obfuscates the values for the changed field:

```
ci360("send", {
  "eventName": "change",
  "attempts": 1,
  "id": "theId",
  "name": "theName",
  "field": {
    "value": "secretValue",
    "type": "text",
    "name": "field1Name",
    "alt": "altVal",
    "changes": 2,
    "id": "field1Id",
    "obscure": "true"
  }
});
```

Click Events

Click events track when someone clicks an element on the page.

Click events do not automatically populate a specific table in the DETAIL data model. Click events can be used only to trigger other events in the user interface, such as steps in a [Business Process event](#).

TIP Unless you need to send a raw event of this type, consider sending a [custom JavaScript event](#) instead. Custom events enable more flexibility because you send any event data as custom attributes. You can then use this data with the corresponding custom JavaScript API event in the user interface.

Click events are not sent unless a [load event](#) has already been triggered. Send a load event before any click event. Otherwise, click events are queued until the next load event is sent.

Here are the parameters for this event type:

Table 5 Parameters for Click Events

Name	Required or Optional	Data Type	Description
eventName	Required	String	The event. The value must be click .
altText	Optional	String	The altText attribute of the click target.
anchorId	Optional	Number, String	The ID of an anchor. When you define an event in the user interface, this value corresponds to Target: id .
anchorName	Optional	String	The name of an anchor. When you define an event in the user interface, this value corresponds to Target: name .
elementTagName	Optional	String	The name of the element that was clicked. When you define an event in the user interface, this value corresponds to Target: tagName .
onClick	Optional	String	The onClick value of the clicked element. When you define an event in the user interface, this value corresponds to Target: onClick .
targetInnerText	Optional	String	The inner text of the element. When you define an event in the user interface, this value corresponds to Target: innerText .
targetSelectorPath	Optional	String	The CSS selector path for the click target. When you define an event in the user interface, this value corresponds to Target: selectorPath .
uri	Optional	String	The destination URI for the event (the HREF of the click target). If a value is not provided, the value for the document.uri variable is set by default. When you define an event in the user interface, this value corresponds to Target: href . Note: To be stored correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'() , ■ reserved characters: ;/? :@=&

Name	Required or Optional	Data Type	Description
------	----------------------------	-----------	-------------

For more information, see the specification for URLs at <https://www.ietf.org/rfc/rfc1738.txt>.

This is an example of a click event:

```
ci360("send", {
  "eventName": "click",
  "elementTagName": "theElementTagName",
  "altText": "click_target_altText",
  "anchorId": "theAnchorId",
  "anchorName": "theAnchorName",
  "targetSelectorPath": "theTargetSelectorPath",
  "targetInnerText": "theTargetInnerText",
  "onClick": "theOnClick"
});
```

Click-Through Events

Click-through events track when a user clicks a spot or creative on a page. Click-through events populate the [SPOT_CLICKED](#) table.

Note: Supplemental information about the creative is recommended when you send this event. This information is used to populate columns such as `task_id`, `spot_id`, and `creative_id`.

- If you are using the SAS tag and the JavaScript Events API, you can send a [getCreatives](#) method to populate data that supplements the event data that you send.
- If you populate creatives through the [Content Request API on page 126](#), you can provide these values in this event's JSON payload.

If the click-through event is marked in a task configuration as a primary or secondary metric, the system generates corresponding conversion or milestone events:

- Conversions are stored in the [CONVERSION_MILESTONE](#) table.
- If the click-through event is marked as a contact or response event, the system generates contact or response events in the corresponding [CONTACT_HISTORY](#) table or [RESPONSE_HISTORY](#) table.

TIP To generate conversions for tasks, ensure that you send an [impression-viewable event](#) before you send a click-through event.

Here are the parameters for this event type:

Table 6 Parameters for Click-Through Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be clickthrough .
spot_nm	Required for content delivered by SAS Customer Intelligence 360	String	For internal use only	The name of the spot that was clicked.
spotId	Required	String	For internal use only	The UUID of the spot that was clicked. This value is in the user interface as part of the URL when you are viewing the spot. For example: <a href="https://<server>/SASWebMarketing/#/spots/3bc28afe-7579-423f-a82d-766cafe57">https://<server>/SASWebMarketing/#/spots/3bc28afe-7579-423f-a82d-766cafe57
uri	Optional	String	SPOT_CLICKED.url_txt	The URL that is associated with the spot that was clicked. If a value is not provided, the value for the document.uri variable is set by default. Note: To be stored correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'() , ■ reserved characters: ;/? :@=& For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt.
taskId	Required for content delivered by the Content Request API	String	SPOT_CLICKED.task_id	UUID of the task for which the user received the content in this spot. Use the corresponding value in the JSON response on page 135 from the Content Request API.

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
creativeld	Required for content delivered by the Content Request API	String	SPOT_CLICKED.creative_id	UUID of the creative returned for this spot. Use the corresponding value in the JSON response on page 135 from the Content Request API.
variantId	Optional and used only for content delivered by the Content Request API	String	For internal use only	For spots that are part of A/B tests, this value represents which variant of the A/B test was delivered. Use the corresponding value in the JSON response on page 135 from the Content Request API.
requestId	Optional and used only for content delivered by the Content Request API	String	SPOT_CLICKED.request_id	Unique ID that identifies the current instance of the impression. This ID is stored in the impression_delivered event. This ID enables you relate spot-related events (SPOT_REQUESTED and SPOT_CLICKED) and impression-related events (IMPRESSION_DELIVERED and IMPRESSION_SPOT_VIEWABLE). Use the corresponding value in the JSON response on page 135 from the Content Request API.
<spot attribute name>	Optional and used only for content delivered by the Content Request API	String	SPOT_CLICKED.properties_map_doc	Name and value for an attribute. These attributes must already be defined for a spot in the Spot Details page in SAS Customer Intelligence 360. For example: <pre>'my_spot1': 'my_spot1_value'</pre>

This is an example of a click-through event:

```
ci360('send', {
  'eventName': 'clickthrough',
  'spotId': 'human readable spot ID',
  'taskId': 'uuid',
```

```

'creativeId': 'uuid',
'variantId': 'uuid',
'requestId': 'uuid',
'<spot_attr_name1>': '<spot_attr_value1>',
...
'<spot_attr_nameN>': '<spot_attr_valueN>'
});

```

Content-Change Events

Content-change events track when the content of a web page changes.

Note: For each page load, the number of content change events is limited to 10. Contact a SAS representative if you want to change this value.

Content-change events do not automatically populate a specific table in the DETAIL data model. Content-change events can be used only to trigger other events in the user interface, such as steps in a [Business Process event](#).

TIP Unless you need to send a raw event of this type, consider sending a [custom JavaScript event](#) instead. Custom events enable more flexibility because you send any event data as custom attributes. You can then use this data with the corresponding custom JavaScript API event in the user interface.

Here are the parameters for this event type:

Table 7 Parameters for Content-Change Events

Name	Required or Optional	Data Type	Description
attributeName	Required	String	The name of an attribute that is associated with the event.
eventName	Required	String	The event. The value must be contentchange .
oldValue	Required	String	The previous value for an element or attribute.
selector	Required	String	The CSS selector information of the target that is associated with the event.
type	Required	String	You must use one of these values: ■ attribute ■ childadd ■ childremove

Name	Required or Optional	Data Type	Description
newValue	Optional	String	The new value for an element or attribute.
id	Optional	Number, String	An ID for the event.
name	Optional	String	The name of the target that is associated with the event.

This is an example of a content-change event:

```
ci360('send', {
  'attributeName': 'attributeName',
  'eventName': 'contentchange',
  'eventUid': 'user_id',
  'newValue': 'thisNewVal',
  'oldValue': 'thisOldVal',
  'id': '123',
  'name': 'theName',
  'selector': 'my>element>selector',
  'type': 'attribute'
});
```

Custom Events

Custom events track any behavior that might not be associated with other event types. Custom events are more flexible than raw events (such as a click event) because they enable you to send any event data as custom attributes.

If you want to use a raw event in SAS 360 Engage, in most cases you should use a custom JavaScript event. Then associate this event with a corresponding [JavaScript API event](#) in the user interface.

Custom events populate both the [CUSTOM_EVENTS](#) table and [GOAL_DETAILS](#) table.

Here are the parameters for this event type:

Table 8 Parameters for Custom Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be custom .
apiEventKey	Required	String	For internal use only	The unique key that is associated with an event defined in the user interface. For more information, see “Using the JavaScript Events API with Tasks” .

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				The event can be associated with a custom JavaScript API event in the user interface. in the user interface.
customName	Optional	String	GOAL_DETAILS.goal_nm CUSTOM_EVENTS.custom_event_nm	This is a predefined field that is for special use cases. Previously, advanced collection rules populated the corresponding column in the database tables. You can specify a value, but it is not required.
customGroupName	Optional	String	GOAL_DETAILS.goal_group_nm CUSTOM_EVENTS.custom_event_group_nm	This is a predefined field that is for special use cases. Previously, advanced collection rules populated the corresponding column in the database tables. You can specify a value, but it is not required.
customRevenue	Optional	Double	GOAL_DETAILS.goal_revenue_amt CUSTOM_EVENTS.custom_revenue_amt	This is a predefined field that is for special use cases. Previously, advanced collection rules populated the corresponding column in the database tables. You can specify a value, but it is not required.
<custom fields>	Optional	Any	GOAL_DETAILS.properties_map_doc CUSTOM_EVENTS.properties_map_doc	Custom fields that record data that you want to send. Field names should start with a letter. In the data tables, the custom event's record contains all the custom fields in one column. Pairs of custom fields are separated by CTRL-C characters, and the key and value for each field are separated by the CTRL-B character. TIP Use a custom prefix for field names to avoid conflicts with field names from built-in parameters. For example, custom fields could be named "ci360_field2".

This is an example of a custom event:

```
ci360("send", {
  "eventName": "custom",
```

```

    "apiEventKey": "myCustomApikey",
    "ci360_customField1": "customVal1",
    "ci360_field2": "val2",
    "ci360_field3": 3
  });

```

Document Events

Document events track when users interact with document types on your site. These interactions are typically document downloads.

Document events populate the [DOCUMENT_DETAILS](#) table.

Here are the parameters for this event type:

Table 9 *Parameters for Document Events*

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be document .
url	Required	String	DOCUMENT_DETAILS.uri_txt	The URL of the document.
altText	Optional	String	DOCUMENT_DETAILS.alt_txt	Alternate text that is related to the document event.
id	Optional	String	DOCUMENT_DETAILS.link_id	An ID for the document event.
name	Optional	String	DOCUMENT_DETAILS.link_name	A name that is associated with the document event.
selectorPath	Optional	String	DOCUMENT_DETAILS.link_selector_path	The CSS selector path that is related to the document.

This is an example of a document event:

```

ci360("send", {
  "eventName": "document",
  "altText": "theAltText",
  "id": "theId",
  "url": "theUrl",
  "name": "theName",
  "selectorPath": "theSelectorPath"
});

```

Identity Events

Attach Identity The attach identity event tracks when a customer identifies themselves (for example, by signing in). Identity events are used to identify users and to determine the correct datahub ID that is stored with events after the identity event (login) is sent.

There are two types of identity events:

- an `attachIdentityBlocking` event, which prevents the SAS tag from sending any subsequent events until the identity event is received. The blocking event ensures that any events that occur immediately after the identity event are associated with the correct datahub ID.

This event type starts with the code `ci360("attachIdentityBlocking", ...)`.

- an `attachIdentity` event, which is sent asynchronously and does not prevent the SAS tag from sending additional events. With this event type, event data that is sent immediately after the event might be associated with the datahub ID that was already established for the session.

This event type starts with the code `ci360("attachIdentity", ...)`.

The mapping between the datahub ID and the login value that you provide populates the `IDENTITY_ATTRIBUTES` table. This event is recorded one time only and not for each login of the same person. Identity events do not populate any records in detail tables.

Here are the parameters for the attach identity event types:

Table 10 Parameters for the Attach Identity Events

Name	Required or Optional	Data Type	Description
<code>loginId</code>	Required	String	The value of the user's login. Make sure that this value corresponds to the value for <code>loginEventType</code> .
<code>loginEventType</code>	Required	String	The type of login event that is being sent. You must use one of these values: ■ <code>customer_id</code> ■ <code>login_id</code> ■ <code>subject_id</code>
<code>obfuscateFields</code>	Optional	Object	Any parameters that should be obfuscated. For this event, the value would most likely contain only one entry for the <code>loginId</code> parameter: <pre>"obfuscateFields": ["loginId"]</pre> Note: The value is a string representation of an array. For more information, see “Obscuring Values in the JavaScript Events API”.

Here is an example of a `attachIdentityBlocking` event that uses a customer ID:

```
ci360("attachIdentityBlocking", {
  "loginId": "customer.name@sas.com",
```

```
"loginEventType": "customer_id"
});
```

Here is an example of the same data but sent as an `attachIdentity` event (non-blocking):

```
ci360("attachIdentity", {
  "loginId": "customer.name@sas.com",
  "loginEventType": "customer_id"
});
```

Here is an example of an `attachIdentityBlocking` event that obfuscates the customer's ID:

```
ci360("attachIdentityBlocking", {
  "loginId": "customer.name@sas.com",
  "loginEventType": "customer_id",
  "obfuscateFields": ["loginId"]
});
```

Detach Identity Events Detach identity events disconnect a user from their current, known identity in the system. When you send this type of event, the system does these actions:

- 1 detaches the current visitor cookie from the datahub ID
- 2 assigns a new identity ID to the visitor cookie

The visitor is then treated as an anonymous user, and all activity is stored with the new datahub ID. The user's new datahub ID is not connected to any identifier (`login_id`, `customer_id`, and so on) until you send an `attachIdentity` call to explicitly define the new user.

There are two types of detach identity events:

- cross-channel detach event. This event type detaches users from their identity in both web sessions and mobile sessions at the same time. This event detaches both the visitor ID (for the web session) and the device ID (for the mobile session) from the datahub ID that is associated with the user.

This is an example of how to send a cross-channel detach event:

```
ci360('detachIdentityCrossChannel');
```

- web-only detach event. This event type detaches users from their identity only in a web session. This is an example of how to send a web-only detach event:

```
ci360('detachIdentity');
```

Detach identity events return a response of `undefined`. In addition, detach identity events do not populate any records in the detail tables or identity tables.

Deprecated: Identity Event

Note: This event is deprecated. Modify any calls to use the [attachIdentity](#) function instead.

Identity events track when a customer identifies themselves (for example, by signing in).

Here are the parameters for this event type:

Table 11 Parameters for Identity Events

Name	Required or Optional	Data Type	Description
eventName	Required	String	The event. The value must be identity .
loginId	Required	String	The value of the identity type.
loginEventType	Required	String	You must use one of these values: ■ account_id ■ customer_id ■ household_id ■ login_id ■ subject_id

This is an example of an identity event:

```
ci360("send", {
  "eventName": "identity",
  "loginId": "customer.name@sas.com",
  "loginEventType": "customer_id"
});
```

Impression Events

Impression (spot change) events track when a creative is delivered by SAS Customer Intelligence 360. Impression events do not typically take into account whether a user actually views the content or interacts with it.

Note: When you deliver content through the Content Request API, this event is generated for you and does not need to be sent explicitly.

Impression events populate the [IMPRESSION_DELIVERED](#) table. Most of the fields in this table (such as TASK_ID, SPOT_ID, CREATIVE_ID, CHANNEL_NM, and so on) are populated automatically based on the spot_name that you send with the event. This information is gathered from the getCreatives call that you send beforehand.

Here are the parameters for this event type:

Table 12 *Parameters for Impression Events*

Name	Required or Optional	Data Type	Description
eventName	Required	String	The event. The value must be impression .
spotName	Required	String	The name of the spot.

This is an example of an impression event:

```
ci360('send', {
  'eventName': 'impression',
  'spotName': 'events_spot1'
});
```

Impression-Viewable Events

Impression-viewable (spot viewable) events track when a creative is displayed in your content. How you define “viewable” can be specific to your needs. For example, you might decide that a creative is viewable when a user scrolls past its position on the page. Impression-viewable events populate the [IMPRESSION_SPOT_VIEWABLE](#) table.

Note: Supplemental information about the creative is recommended when you send this event. This information is used to populate columns such as `task_id`, `spot_id`, and `creative_id`.

- If you are using the SAS tag and the JavaScript Events API, you can send a [getCreatives](#) method to populate data that supplements the event data that you send.
- If you populate creatives through the [Content Request API on page 126](#), you can provide these values in this event’s JSON payload.

An impression-viewable event is required to identify conversions in the user interface. To generate conversions for tasks, make sure that you send an impression-viewable event before you send a [click-through event](#). Conversions are counted for users against a task only when the users fulfill these requirements:

- 1 Saw the content, and therefore generated an `impression_viewable` record.
- 2 Triggered the event that is defined as a primary or secondary metric in the corresponding task.

Here are the parameters for this event type:

Table 13 Parameters for Impression-Viewable Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be impressionViewable .
spot_nm	Required for content delivered by SAS Customer Intelligence 360	String	For internal use only	The name of the spot.
spotId	Required	String	For internal use only	The UUID of the spot. This value is in the user interface as part of the URL when you are viewing the spot. For example: <a href="https://<server>/SASWebMarketing/#/spots/3bc28afe-7579-423f-a82d-766cafe57">https://<server>/SASWebMarketing/#/spots/3bc28afe-7579-423f-a82d-766cafe57
taskId	Required for content delivered by the Content Request API	String	IMPRESSION_SPOT_VIEWABLE.task_id	UUID of the task for which the user received the content in this spot. Use the corresponding value in the JSON response on page 135 from the Content Request API.
creativeId	Required for content delivered by the Content Request API	String	IMPRESSION_SPOT_VIEWABLE.creative_id	UUID of the creative returned for this spot. Use the corresponding value in the JSON response on page 135 from the Content Request API.
variantId	Optional and used only for content	String	For internal use only	For spots that are part of A/B tests, this value represents which variant of the A/B test was delivered. Use the corresponding value in the JSON response on page 135 from the Content Request API.

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
	delivered by the Content Request API			
requestId	Optional and used only for content delivered by the Content Request API	String	IMPRESSION_SPOT_VIEWABLE.request_id	Unique ID that identifies the current instance of the impression. This ID is stored in the impression_delivered event. This ID enables you relate spot-related events (SPOT_REQUESTED and SPOT_CLICKED) and impression-related events (IMPRESSION_DELIVERED and IMPRESSION_SPOT_VIEWABLE).
<spot attribute name>	Optional and used only for content delivered by the Content Request API	String	IMPRESSION_SPOT_VIEWABLE.properties_map_doc	Name and value for an attribute. These attributes must already be defined for a spot in the Spot Details page in SAS Customer Intelligence 360. For example: <pre>'my_spot1': 'my_spot1_value'</pre>

This is an example of an impression-viewable event:

```
ci360('send', {
  'eventName': 'impressionViewable',
  'spotId': 'uuid',
  'taskId': 'uuid',
  'creativeId': 'uuid',
  'variantId': 'uuid',
  'requestId': 'uuid',
  '<spot_attr_name1>': '<spot_attr_value1>',
  ...
  '<spot_attr_nameN>': '<spot_attr_valueN>'
});
```

Internal-Search Events

Internal-search events track users' interaction with the search functionality of your site.

Internal-search events populate the [SEARCH_RESULTS](#) table.

Here are the parameters for this event type:

Table 14 *Parameters for Internal-Search Events*

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be internalSearch .
apiEventKey	Required	String	For internal use only	The unique key that is associated with an event defined in the user interface. For more information, see "Using the JavaScript Events API with Tasks" . The event can be associated with an Internal Search event in the user interface.
fieldId	Required	String	SEARCH_RESULTS.srch_field_id	The ID of the search field on your site.
fieldName	Required	String	SEARCH_RESULTS.srch_field_name	The name of the search field on your site.
resultsDisplayedFlag	Required	String	SEARCH_RESULTS.results_displayed_flag	Whether the search results were displayed. The value must be the string true or false . When the value is true , an additional record is created in the SEARCH_RESULTS_EXT table .
searchName	Required	String	SEARCH_RESULTS.search_nm	The name that is associated with the search.
term	Required	String	SEARCH_RESULTS.srch_phrase	The term that was used in the search.
resultsDisplayed	Optional	Integer	SEARCH_RESULTS.search_results_displayed	The number of search results that is returned.

This is an example of an internal-search event:

```
ci360("send", {
  "eventName": "internalSearch",
  "apiEventKey": "myInternalSearchKey",
  "searchName": "theSearchName",
  "term": "theSearchTerm",
  "resultsDisplayed": 5,
  "resultsDisplayFlag": "true",
  "fieldId": "theFieldId",
```

```
"fieldName": "theFieldName"
});
```

JavaScript Variable-Change (jsvarchange) Events

JavaScript variable-change (jsvarchange) events track when a JavaScript variable changes on a web page.

Note: For each page load, the number of JavaScript variable changes is limited to 10. Contact a SAS representative if you want to change this value.

JavaScript variable-change events do not automatically populate a table in the DETAIL data model. These events can be used only to define other events in the user interface, such as steps in a [Business Process event](#).

TIP Unless you need to send a raw event of this type, consider sending a [custom JavaScript event](#) instead. Custom events enable more flexibility because you send any event data as custom attributes. You can then use this data with the corresponding custom JavaScript API event in the user interface.

Here are the parameters for this event type:

Table 15 Parameters for JavaScript Variable-Change (jsvarchange) Events

Name	Required or Optional	Data Type	Description
eventName	Required	String	The event. The value must be <code>jsvarchange</code> .
name	Required	String	The name of the JavaScript variable.
newValue	Required	String	The new value of the JavaScript variable.
oldValue	Required	String	The previous value of the JavaScript variable.
action	Optional	String	The type of action. You must use one of these values: ■ <code>add</code> ■ <code>update</code> If the parameter is not specified, the default value is <code>update</code>.
uri	Optional	String	The URI that is associated with the changed JavaScript variable. If a value is not provided, the value for the <code>document.uri</code> variable is set by default.

Name	Required or Optional	Data Type	Description
Note: To be stored correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'(), ■ reserved characters: ;/?:@=& For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt.			

This is an example of a JavaScript variable-change event:

```
ci360("send", {
  "eventName": "jsvchange",
  "name": "jsvchangeName",
  "action": "update",
  "oldValue": "oldVal",
  "newValue": "newVal"
});
```

Load Events

Load events track when a page load happens. To use a load event in SAS 360 Engage (for example, in targeting), select a standard Page View event in the user interface. You can select attributes such as the page domain, title, and referrer. These values are available for all load events, regardless of whether they are collected automatically or sent explicitly through the API.

Note: If you want to send a [click event](#) through the JavaScript Events API, you must send a load event to populate data for the page. Otherwise, click events are queued until a load event is sent.

Load events populate the [PAGE_DETAILS](#) table.

Here are the parameters for this event type:

Table 16 Parameters for Load Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be load .
pageTitle	Optional	String	PAGE_DETAILS.page_desc	The page title. If the value is not specified, the source document's title is used by default.

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
referrer	Optional	String	PAGE_DETAILS.referrer_url_txt	The HTTP referrer. If the value is not specified, the source document's referrer is used by default.
uri	Optional	String	PAGE_DETAILS.page_url_txt	The URI of the load event. If a value is not provided, the value for the document.uri variable is set by default. Note: To be stored correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'(), ■ reserved characters: ;/?:@=& For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt .

This is an example of a load event:

```
ci360("send", {
  "eventName": "load",
  "uri": "loadEventUri",
  "pageTitle": "thePageTitle",
  "referrer": "theReferrer"
});
```

Media Events

Media events track when a customer interacts with a media element.

Note: For each page load, the number of media events is limited to 200.

Media events populate the [MEDIA_ACTIVITY_DETAILS](#) table. The system also calculates values and populates records in the [MEDIA_DETAILS](#) table and the [MEDIA_DETAILS_EXT](#) table when media events occur in these situations:

- in the same session
- on the same page
- at the same URL (which means the same media file was played)

Here are the parameters for this event type:

Table 17 *Parameters for Media Events*

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
action	Required	String	MEDIA_ACTIVITY_DETAILS.action	The action that is associated with the media event. You must use one of these values: ■ end ■ start ■ stop
duration	Required	Number, String	MEDIA_DETAILS.media_duration_seconds	The length of the media. The exact format and value of this attribute depend on the values that are returned by the media player. The value must be an integer or a decimal.
eventName	Required	String	For internal use only	The event. The value must be media .
id	Required	String		The ID of the media player's element.
player	Required	String	MEDIA_DETAILS.media_player_name	A string or number that identifies the media player. The maximum length for this value is 30 characters.
position	Required	Number, String	MEDIA_ACTIVITY_DETAILS.playhead_position	The current position of the playback controller. The exact format depends on the values that are returned by the media player. The value must be an integer.
url	Required	String	MEDIA_ACTIVITY_DETAILS.media_uri_txt MEDIA_DETAILS.media_uri_txt MEDIA_DETAILS_EXT.media_uri_txt	A URL that is associated with the media event. The maximum length for this value is 260 characters.

This is an example of a media event:

```
ci360("send", {
  "eventName": "media",
  "id": "elementId",
  "url": "http://www.someSource.com/mySource",
```

```

    "player": "YouTube",
    "position": "0",
    "duration": "65",
    "action": "start"
  });

```

Product View Events

Product View events track when a customer views a product on your site.

Product view events populate the [PRODUCT_VIEWS](#) table.

Here are the parameters for this event type:

Table 18 Parameters for Product View Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be productView .
apiEventKey	Required	String	For internal use only	The unique key that is associated with an event defined in the user interface. For more information, see “Using the JavaScript Events API with Tasks” . This event can be associated with a Product View event in the user interface.
id	Required	String	PRODUCT_VIEWS. product_id	The ID of the product. You must specify at least one of the following parameters: ■ id ■ name ■ sku
name	Required	String	PRODUCT_VIEWS. product_nm	The name of the product. You must specify at least one of the following parameters: ■ id ■ name ■ sku
sku	Required	String	PRODUCT_VIEWS. product_sku	The SKU of the product. You must specify at least one of the following parameters: ■ id ■ name

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				■ sku
availability Message	Optional	String	PRODUCT_VIEWS.availability_message_txt	A message that is related to item availability. The maximum length for this value is 650 characters. For example, the value could be back-ordered .
group	Optional	String	PRODUCT_VIEWS.product_group_nm	A group that the cart item belongs to. The maximum length for this value is 130 characters. For example, the value could be sporting goods .
unitPrice	Optional	Number	PRODUCT_VIEWS.price_val	The price for a single unit of the cart item. This value must be an integer or a decimal.
savingsMessage	Optional	String	PRODUCT_VIEWS.saving_message_txt	A message that is related to discounts. The maximum length for this value is 650 characters.
shippingMessage	Optional	String	PRODUCT_VIEWS.shipping_message_txt	A message that related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.

This is an example of a Product View event:

```
ci360("send", {
  "eventName": "productView",
  "apiEventKey": "myProdViewEventKey",
  "id": "theId",
  "name": "theName",
  "sku": "theSku",
  "unitPrice": "theUnitPrice",
  "group": "theGroup",
  "availabilityMessage": "theAvailabilityMessage",
  "shippingMessage": "theShippingMessage",
  "savingsMessage": "theSavingsMessage"
});
```

Promotion Events

Promotion events track when users interact with your site's in-site advertising, offers, and campaigns. Customer interaction can be anything from a simple click-through to something else such as using a promotion code during checkout.

Promotion events populate the [PROMOTION_DISPLAYED](#) table and the [PROMOTION_USED](#) table. The `record_type` attribute determines whether it is a display event or a use event (which is a click-through), and records are written to the appropriate table.

Here are the parameters for this event type:

Table 19 *Parameters for Promotion Events*

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be <code>promotion</code> .
apiEventKey	Required	String	For internal use only	The unique key that is associated with an event defined in the user interface. For more information, see “Using the JavaScript Events API with Tasks” . This event can be associated with a Promotion event in the user interface.
name	Required	String	PROMOTION_DISPLAYED.promotion_nm PROMOTION_USED.promotion_nm	The promotion's name.
recordType	Required	String	Determines which tables the records are written to	How a user interacted with a promotion. For example, you could use these values: ■ display: indicates that a promotion was shown to a user. ■ use: indicates that a user interacted with a promotion.
type	Required	String	PROMOTION_DISPLAYED.promotion_type_nm PROMOTION_USED.promotion_type_nm	The promotion's type. Some examples of types are banner ads, organic search, pay-per-click, referrals, partner sites, affiliates, email, and newspaper articles.
creativeName	Optional	String	PROMOTION_DISPLAYED.promotion_creative_nm PROMOTION_USED.promotion_creative_nm	The name of the creative that is associated with the promotion.

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
placementId	Optional	String	PROMOTION_DISPLAYED.promotion_placement_nm PROMOTION_USED.promotion_placement_nm	An ID where the promotion's banner or link is located.
trackingCode	Optional	String	PROMOTION_DISPLAYED.promotion_tracking_cd PROMOTION_USED.promotion_tracking_cd	A campaign tracking code that is used to link to additional campaign information (such as cost, media agency used, and so on) in an external table.

This is an example of a promotions event:

```
ci360('send', {
  "eventName": "promotion",
  "apiEventKey": "myPromotionEventKey",
  "trackingCode": "theTrackingCode",
  "creativeName": "theCreativeName",
  "placementId": "thePlacementId",
  "name": "theName",
  "recordType": "display",
  "type": "theType"
});
```

Submit Events

Submit events track when a customer submits a form. Submit events should include names, values, and number of submission attempts from at least one field in the form.

Submit events populate the [FORM_DETAILS](#) table. Every field that is listed in a `fields{}` object is written as a separate record in the `FORM_DETAILS` table.

Submit events can also be used in the user interface to derive other events such as steps in a [Business Process event](#).

TIP Unless you need to send a raw event of this type, consider sending a [custom JavaScript event](#) instead. Custom events enable more flexibility because you send any event data as custom attributes. You can then use this data with the corresponding custom JavaScript API event in the user interface.

Here are the parameters for this event type:

Table 20 Parameters for Submit Events

Name	Required or Optional	Data Type	Table Location (as Table.Column)	Description
eventName	Required	String	For internal use only	The event. The value must be submit .
fields	Required	Object	The table fields that are populated depend on the item that is sent: ■ name is sent to FORM_DETAILS.form_nm. ■ value is sent to FORM_DETAILS.form_nm_value. ■ changes is sent to FORM_DETAILS.change_index_no.	An object that defines one or more form fields. Fields are defined like this example: <pre>"fields": [{ "name": "field1Name", "value": "field1Value", "changes": 1 }, { "name": "field2Name", "value": "field2Value", "changes": 1 }]</pre> Note: A field must contain the following parameters: name, value, and changes.
id	Required	String	FORM_DETAILS.form_nm	The ID of the form. Note: You must send either the <code>id</code> parameter or the <code>name</code> parameter, but both parameters are not required. If both parameters are sent, the <code>id</code> parameter is used to populate the FORM_NM column.
name	Required	String	FORM_DETAILS.form_nm	The name of the submitted form. Note: You must send either the <code>id</code> parameter or the <code>name</code> parameter, but both parameters are not required. If both parameters are sent, the <code>id</code> parameter is used to populate the FORM_NM column.
attempts	Optional	Number	FORM_DETAILS.attempt_index_cnt	The number of attempts that has been made to submit the field value. This value must be an integer. If a value is not provided, the default is set to 1.

This is an example of a submit event that includes fields as custom parameters:

```
ci360("send", {
  "eventName": "submit",
  "id": "theId",
```

```

"name": "theName",
"attempts": 1,
"fields": [
{
  "value": "field1Val",
  "name": "field1Name",
  "changes": 3,
  "id": "field1Id"
},
{
  "value": "customerLogin",
  "name": "login",
  "changes": 1,
  "id": "username"
}
]);

```

This example obscures the value of a login value:

```

ci360("send", {
  "eventName": "submit",
  "location": "theLocation",
  "id": "theId",
  "name": "theName",
  "attempts": 1,
  "fields": [
    {
      "value": "customerLogin",
      "name": "login",
      "changes": 1,
      "id": "username",
      "obscure": "true"
    }
  ]
});

```

Visibility-Change Events

Visibility changes occur when a user navigates away from a page or comes back. For example, if a user is on sas.com but then navigates to a different page, you can send a visibility-change event with a name/value pair of `hidden=true`. When the user comes back to sas.com, you can send a visibility-change event with a name/value pair of `hidden=false`.

These events are used to generate records in the [PAGE_DETAILS_EXT](#) table.

Here are the parameters for this event type:

Table 21 Parameters for Visibility-Change Events

Name	Required or Optional	Data Type	Description
eventName	Required	String	The name of the event. This value must be <code>visibilityChange</code> .

Name	Required or Optional	Data Type	Description
uri	Required	String	The URI for the page where the visibility-change event occurred. Note: To be stored correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'(), ■ reserved characters: ;/?:@=& For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt.
hidden	Required	Boolean	Specifies whether the user moved away from the page (hidden=false) or made the page active (hidden=true).

This is an example of a visibility-change event:

```
ci360("send", {
  "eventName": "visibilityChange",
  "uri": "https://www.mySite.com/products/shoes#hiTops",
  "hidden": false
});
```

Get Spot Definitions

Use this call to find out which spots are on a page and to ask the system what content should be injected. You can use this information to retrieve the relevant task, spot, creative, and variant IDs and pass this information to events (such as impression events or impression viewable events).

This is an example of how to use the JavaScript Events API to get a spot definition:

```
ci360("getSpotDefinitions", {
  domain: "<site domain>",
  path: "<site path>",
  params: {
    <param1 name>: "<URL param1 value>",
    <param2 name>: "<URL param2 value>"
  },
  names: [<spot name>]
}, <callback>);
```

Note: The `getSpotDefinitions()` function is asynchronous, so you must use a callback function to process the result.

Here are the parameters for this API call:

Table 22 *Parameters for a getSpotDefinitions Call*

Name	Required or Optional	Data Type	Description
domain	Required	String	The base domain of the site.
path	Optional	String	The full path of the site.
params	Optional	Object	An object that contains one or more URL parameters and their values. For example: <pre>{ pathLevel1: "myApp", pathLevel2: "myPage" }</pre>
names	Optional	Array of Strings	An array that contains one or more spot names. For example: <pre>["mySpot1", "mySpot2", "mySpot3"]</pre>
callback	Required	Function	The callback function. The callback should include parameters for an error and the returned data. For example, this function includes a callback: <pre>ci360("getSpotDefinitions", { domain: "my.domain.com" }, function (err, data) { // do something with returned data });</pre> The returned data is a Map object that associates spots' names with their ID, selector, and any attributes.

Get Creative Types

About the Calls for Creatives

Use the getCreatives call or the getPersonalizedCreative call to retrieve a creative from SAS Customer Intelligence 360 and populate a spot with that content.

- Use the getCreative call when you want a personalized creative but the default creative is acceptable for the spot. If SAS Customer Intelligence 360 cannot return a personalized creative before the time-out period, the system returns the default creative for the spot.
- Use the getPersonalizedCreative call when you want to force the system to return a personalized creative. This call continues to poll the system until a personalized creative (and not the default

creative) is returned. This call would be useful in situations where targeting criteria might have been met but is still being processed by SAS Customer Intelligence 360.

To personalize a creative, the visitor ID is used to determine the customer's identity. Ideally, you should trigger an identity event at least once for a browser (and possibly once per session). Then, any subsequent activity (such as impressions, click-through events, and so on) is associated with that identity.

Get Creatives

This is an example of how to use the JavaScript Events API to get a creative:

```
ci360("getCreatives", {
  domain: "<site domain>",
  path: "<site path>",
  params: {
    <param1 name>: "<URL param1 value>",
    <param2 name>: "<URL param2 value>"
  },
  attributes: {
    "<spot name>": {<spot attribute name>: "<spot attribute value>"}
  },
  names: [<"spot name">]
}, <callback>);
```

Note: The `getCreatives()` function is asynchronous, so you must use a callback function to process the result.

Also, there is only one system object that holds the information about the spot and task. When you send a `getCreatives` call, it overwrites any data from the last call with the latest information.

Here are the parameters for this API call:

Table 23 Parameters for a `getCreatives` Call

Name	Required or Optional	Data Type	Description
domain	Required	String	The base domain of the site.
path	Optional	String	The full path of the site.
params	Optional	Object	An object that contains one or more URL parameters and their values. For example: <pre>{ pathLevel1: "myApp", pathLevel2: "myPage" }</pre>
attributes	Optional	Object	An object that contains one or more spot objects. A spot object consists of a spot name and some attributes. For example:

Name	Required or Optional	Data Type	Description
			<pre> { mySpot1: { spotAttr1: "mySpot1 attr1 value", spotAttr2: "mySpot1 attr2 value" }, mySpot2: { spotAttr1: "mySpot2 attr1 value", spotAttr2: "mySpot2 attr2 value" } } </pre>
names	Optional	Array of Strings	An array that contains one or more spot names. For example: <pre>["mySpot1", "mySpot2", "mySpot3"]</pre>
callback	Required	Function	The callback function. The callback should include parameters for an error and the returned data. For example, this function includes a callback: <pre> ci360("getCreatives", { domain: "my.domain.com" }, function (err, data) { // do something with returned data }); </pre> The returned data is a Map object that associates spots' names with objects that contain the HTML source, creative ID, task ID, and variant ID.

Get a Personalized Creative

This is an example of how to use the JavaScript Events API to get a personalized creative:

```

ci360("getPersonalizedCreative", {
  domain: "<site domain>",
  path: "<site path>",
  params: {
    <param1 name>: "<URL param1 value>",
    <param2 name>: "<URL param2 value>"
  },
  attributes: {
    "<spot name>": {<spot attribute name>: "<spot attribute value>"}
  },
  name: "<spot name>",
  maxTries: <maximum number of attempts>,
  delay: <delay in milliseconds>
}, <callback>);

```

Note: The `getPersonalizedCreative()` function is asynchronous, so you must use a callback function to process the result.

Here are the parameters for this API call:

Table 24 Parameters for a *getPersonalizedCreative* Call

Parameter	Required or Optional	Data Type	Description
domain	Required	String	The base domain of the site.
path	Optional	String	The full path of the site.
params	Optional	Object	An object that contains one or more URL parameters and their values. For example: <pre> { pathLevel1: "myApp", pathLevel2: "myPage" } </pre>
attributes	Optional	Object	An object that contains one or more spot objects. A spot object consists of a spot name and some attributes. For example: <pre> { mySpot1: { spotAttr1: "mySpot1 attr1 value", spotAttr2: "mySpot1 attr2 value" }, mySpot2: { spotAttr1: "mySpot2 attr1 value", spotAttr2: "mySpot2 attr2 value" } } </pre>
name	Optional	String	A spot name. For example: "mySpot1"
maxTries	Optional	Number	The maximum number of times to request a personalized creative from server. The default value is 1. The maximum value must be less than or equal to 1000. Note: If you set the value too high, you could unintentionally extend the user's session or overwhelm the server.
delay	Optional	Number	The delay between retries. The default value is 1000. The minimum value must be greater than or equal to 100.
callback	Required	Function	The callback function. The callback should include parameters for an error and the returned data. For example, this function includes a callback: <pre> ci360("getPersonalizedCreative", { </pre>

Parameter	Required or Optional	Data Type	Description
			<pre> domain: "my.domain.com" }, function (err, data) { // do something with returned data }); </pre> The returned data is an object that contains the HTML source, creative ID, task ID, and variant ID. The returned data is <code>null</code> if the creative has not been personalized.

Using the JavaScript Events API, Version 1

Note: This version of the JavaScript Events API will be deprecated in a future release. Migrate your site to use [“Using the JavaScript Events API, Version 2”](#) when it is possible.

SAS support can assist with technical issues for the SAS tag and the JavaScript Events API, but they cannot provide guidance on website design. Support is limited to the functionality of the SAS tag and the related features.

The JavaScript Events API enables you to define and send events to SAS 360 Discover and identity events to SAS 360 Foundations. This API gives you full control over which events to send and when to send them. This level of control is in contrast to the default configuration of the SAS tag, which sends a default set of events automatically. The JavaScript Events API is particularly useful when you are working with single-page applications (created with React, Angular, or similar frameworks).

Configure the JavaScript Events API

Send an Event

This is an example of how to send a basic event with two parameters:

```

com_sas_ci_acs.ap.i("event", "send",
{
  "event": "<event_type>",
  "attribute_1": "my_value1", // set as a name/value pair
  "attribute_2": {           // set as a name/object pair
    "value": "my_value2",
    "obscured": "true"
  }
}
);

```

Then, create an event object in JavaScript with parameters and values for the event. The parameters that you define depend on the type of event that you send:

- Each event type has type-specific parameters.

- You can add custom parameters if you need to.
- Parameters can be set as name/value pairs or as name/object pairs.
- The value for every attribute can be obscured. To obscure a value, you set the attribute as a name/object pair and add the “obscured” attribute to the object’s members.

Note: The number of events that is collected for each session is limited to a maximum of 2,000.

When you submit an event, the JavaScript Events API returns “success” to the logging console (or standard output stream). If the event has errors, the API returns the validation error.

(Optional) Wait for the JavaScript Events API before Sending an Event

Before you send an event, the JavaScript Events API must be loaded. In many environments, the API loads quickly and would be available. However, to be safe you can use a wrapper function to ensure that the API is loaded. The wrapper function makes sure that the `com_sas_ci_acs.ap` object is defined, and sends an event only when it is.

This JavaScript code is an example of a wrapper function that waits for the API to become available:

```
let tries = 0; // This is a counter of how many attempts are made.
const MAX_TRIES = 5; // This is the maximum attempts to make. Modify this value
                      // to make more attempts.
const TIMEOUT_MS = 1000; // The amount of time between retries, in milliseconds.

function callWhenV1ApiReady(callbackFunction) {
  if (typeof callbackFunction === 'function') {
    if (typeof com_sas_ci_acs !== 'undefined' &&
        typeof com_sas_ci_acs.ap !== 'undefined' &&
        typeof com_sas_ci_acs.ap.i !== 'undefined') {
      callbackFunction();
      tries = 0;
    }
    else {
      if (tries < MAX_TRIES) {
        tries++;
        setTimeout(callWhenV1ApiReady, TIMEOUT_MS, callbackFunction);
      }
    }
  }
}
```

For each event that you want to send, send the event as a function parameter inside the `callWhenV1ApiReady()` function:

```
callWhenV1ApiReady(function () {
  com_sas_ci_acs.ap.i("event", "send",
  {
    "event": "<event_type>",
    "attribute_1": "my_value1", // set as a name/value pair
    "attribute_2": {           // set as a name/object pair
      "value": "my_value2",
      "obscured": "true"
    }
  }
  });
});
```

Event Types

Cart Events

Cart events contain general information about a cart such as the total value, tax, shipping cost, payment type, delivery type, and other shipping and billing information.

Table 25 *Parameters for Cart Events*

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be <code>cart</code> .
cart_type	Required	String	The type of cart. Use one of these values: ■ <code>cartview</code> ■ <code>checkout</code> ■ <code>purchase</code>
cart_id	Required	String, Number	The customer's unique cart ID. The value must be less than or equal to 42 characters. When a customer creates a cart, completes a purchase, or cancels a cart, the same cart ID is associated with all of these actions. The <code>cart_id</code> can be used externally to associate this cart with your external commerce system. Make sure that <code>cart_id</code> values are unique for each customer's cart. REST API calls bypass the internal verification to ensure unique IDs for carts, so give each cart a unique ID to avoid duplicate data records. Note: You must provide a <code>cart_id</code> value, an <code>order_id</code> value, or both.
order_id	Required	String, Number	The customer's unique order ID. When a customer completes an order, the order ID is the unique identifier. Make sure that <code>order_id</code> values are unique for each customer's order. REST API calls bypass the internal verification to ensure unique IDs for orders, so give each order a unique ID to avoid duplicate data records. Note: You must provide a <code>cart_id</code> value, an <code>order_id</code> value, or both.
cart_name	Required	String	The name of the cart.

Name	Required or Optional	Type	Description
items	Required	Object	A collection of cart items in the cart. For a list of parameters, see Table 26. This parameter is required, but you can send an empty object if you do not need to send cart items. This is an empty items object: <pre>items: []</pre>
billing_city	Optional	String	The city where the invoice should be sent. The value must be less than or equal to 390 characters.
billing_country	Optional	String	The country where the invoice should be sent. The value must be less than or equal to 85 characters.
billing_postcode	Optional	String	The billing code where the invoice should be sent. The value must be less than or equal to 10 characters. For example, in the United States this could be the ZIP code.
billing_region	Optional	String	The region where the invoice should be sent. The value must be less than or equal to 256 characters. For example, in the United States this could be the state.
delivery_type	Optional	String	The delivery type. The value must be less than or equal to 42 characters. For example, the value might be “USPS” or “FEDEX”.
payment_type	Optional	String	The payment type. The value must be less than or equal to 42 characters. For example, the value might be “VISA” or “CHECK”.
shipping_city	Optional	String	The city where the cart is being shipped. The value must be less than or equal to 390 characters.
shipping_cost	Optional	Number	The cost of shipping. The value must be an integer or decimal.
shipping_country	Optional	String	The country where the cart is being shipped. The value must be less than or equal to 85 characters.
shipping_postcode	Optional	String	The shipping code where the cart is being shipped. The value must be less than or equal to 10 characters. For example, in the United States this could be the ZIP code.

Name	Required or Optional	Type	Description
shipping_region	Optional	String	The region where the cart is being shipped. The value must be less than or equal to 256 characters. For example, in the United States this could be the state.
tax	Optional	Number	The tax amount. The value must be an integer or a decimal.
total_cart_value	Optional	Number	The total amount for the cart. The value must be an integer or a decimal.

Cart items contain product information for each row in a cart. Product information includes these item details:

Table 26 *Parameters for Cart Items*

Name	Required or Optional	Type	Description
product_id	Optional	String, Number	The product ID. The value must be less than or equal to 130 characters. Note: You must provide at least one of these parameters: ■ product_id ■ product_name ■ product_sku
product_name	Optional	String	The product name. The value must be less than or equal to 130 characters.
product_sku	Optional	String, Number	The SKU value for the product. The value must be less than or equal to 100 characters.
product_quantity	Optional	Number	The product quantity. The value must be an integer.
product_unit_price	Optional	Number	The unit price of the product. The value must be an integer or a decimal.
product_group	Optional	String	The product group's name. The value must be less than or equal to 130 characters.
availability_message	Optional	String	A message related to item availability (for example, "back-ordered"). The value must be less than or equal to 650 characters.

Name	Required or Optional	Type	Description
shipping_message	Optional	String	A message related to the shipping type, process, or special instructions. The value must be less than or equal to 650 characters.
savings_message	Optional	String	A message related to discounts. The value must be less than or equal to 650 characters.

This is an example of a cart event that contains two cart items:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "cart",
  "cart_type": "checkout",
  "cart_id": "02",
  "order_id": "102",
  "cart_name": "test_cart",
  "billing_city": "Raleigh",
  "billing_country": "US",
  "billing_postcode": "27605",
  "billing_region": "NC",
  "items" : [
    {
      "product_name": "moon boots",
      "product_sku": "344343234",
      "product_quantity": "1",
      "product_unit_price": "35",
      "product_group": "novelty shoes",
      "availability_message": "available",
      "savings_message": "25off"
    },
    {
      "product_name": "penny loafers",
      "product_sku": "344345687",
      "product_quantity": "1",
      "product_unit_price": "65",
      "product_group": "dress shoes",
      "availability_message": "available"
    }
  ]
});
```

Cart-Action Events

Cart-action events track actions related to adding, removing, and updating items in a cart.

Note: When you create a cart-action event, set parameters that are consistent with your external commerce system. It is important to coordinate these parameters with other systems so that analysis and reports are consistent with what you expect.

Table 27 Parameters for Cart-Action Events

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be cartaction .
action	Required	String	The cart action that is performed. Use one of these values: ■ add ■ addFailed ■ update ■ remove Note: When you specify the remove action, the cart_type attribute is required.
cart_type	Optional	String	The cart type associated with this action. Use one of these values: ■ cartview ■ checkout Note: This attribute is required when the action value is remove.
product_id	Optional	String, Number	The ID for the product. The value must be less than or equal to 130 characters. Note: You must provide at least one of these parameters: ■ product_id ■ product_name ■ product_sku
product_name	Optional	String	The name of the product. The value must be less than or equal to 130 characters.
product_sku	Optional	String, Number	The SKU value for the product. The value must be less than or equal to 100 characters.
cart_id_internal	Optional	String, Number	The cart ID. The value must be less than or equal to 42 characters. Note: You must include at least one of these parameters: ■ cart_id_internal ■ cart_name
cart_name	Optional	String	The cart name.

This is an example of a cart-action event:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "cartaction",
  "product_id": "MN0032",
  "product_sku": "344343234",
  "product_name": "moon boots",
  "action": "add",
  "cart_id_internal": "02",
  "cart_name": "shoes"
});
```

Click Events

About Click Events Click events are used for a few scenarios:

- to collect a click event that cannot be collected by default.

For example, you might have a [custom JavaScript API event](#) defined for click events with a Target:id attribute that starts with "callback". There could be 10 links that are collected automatically by the system and two that are not. For those two links that are not monitored, you could send a click event through the API. This click event could mimic a system click event and provide the necessary attributes for the Goal definition.

- to send a click event that is customized for an advanced collection rule or event in the user interface (such as a business process, promotion, or custom attribute rule). In this case, you use the JavaScript Events API to send a click event, but you define an advanced collection rule or event in the user interface that receives the data from the API.

When you define the associated event or collection rule, make sure the attributes that you define match the parameters that are sent with the click event. The table [Table 28](#) describes the most-used parameters and the attributes in the user interface that they map to.

TIP In addition to the parameters in the table, you can send any other parameters as needed. Parameter names should match attributes that are present in an advanced collection rule or event that is defined in the user interface.

Here are the parameters for this event type:

Table 28 *Parameters for Click Events*

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be <code>click</code> .
anchor_id	Optional	String, Number	The ID of an anchor. This parameter maps to the Target:id attribute in the user interface.
anchor_name	Optional	String	The name of an anchor.

Name	Required or Optional	Type	Description
			This parameter maps to the Target:name attribute in the user interface.
element_tag_name	Optional	String	The name of the element that was clicked. This parameter maps to the Target:tagName attribute in the user interface.
location	Optional	String	The location of the element that was clicked. For example, this value could be coordinates on the page. This parameter can be useful when mimicking a system-generated click event. Note: To be stored correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'() , ■ reserved characters: ;/? :@=& For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt.
on_click	Optional	String	The onClick value. This parameter maps to the Target:onclick attribute in the user interface.
port	Optional	String	The port used on the page. This parameter can be useful when mimicking a system-generated click event.
protocol	Optional	String	The transport protocol for the page. The default value is <code>http</code>. This parameter can be useful when mimicking a system-generated click event.
referrer	Optional	String	The referral information. This parameter can be useful when mimicking a system-generated click event.
target_alt_text	Optional	String	The value of an element's alt text attribute. This parameter maps to the Target:alt attribute in the user interface.
target_inner_text	Optional	String	The inner text of the element.

Name	Required or Optional	Type	Description
			This parameter maps to the Target:innerText attribute in the user interface.
target_selector_path	Optional	String	The selector path for the target. This parameter maps to the Target:selectorPath attribute in the user interface.
title	Optional	String	The title of the page. This parameter can be useful when mimicking a system-generated click event.
<custom attributes>	Optional		A name/value pair that defines any additional data that you want to send. To use these parameters in the user interface, the parameter names must follow these guidelines: ■ The parameter must start with "data-". For example, a parameter name could be data-arrival-airport. ■ The parameter must contain only alphanumeric characters, hyphens (-), and spaces. These parameters map to a target:HTML5 custom data attribute in the user interface.

Note: The URI is sent automatically and is set to the URL of the current page. The URI value maps to the target:href attribute in the user interface.

Example: Click Event Used as a Custom Event In this example, you want to use the JavaScript Events API to send a click event that records information about a flight booking.

In this case, you decide to map a flight's information to a click event's basic parameters. For example, the click event might look like this:

```
com_sas_ci_acs.ap.i("event", "send", {
  "event": "click",
  "anchor_id": "flightbooking",           // this is a fixed string that will
                                          // be used in the advanced collection
                                          // rule
  "anchor_name": "RDU",                  // the departure airport
  "target_alt_text": "LHR",              // the arrival airport
  "target_selector_path": "SAS Airlines", // the airline
  "element_tag_name": "2019-12-25",      // the departure date
  "on_click": "18:05",                   // the departure time
  "target_inner_text": "Economy"         // the class of travel
});
```

In the user interface, you would then define a [custom JavaScript API event](#) in **Advanced Collection Rules**. In the **Page Event** section, complete these steps:

- 1 Select **Click** from the **Event** menu.
- 2 Add the Target:id attribute and specify that the attribute's value must equal **flightbooking**. This action ensures you are receiving data from the correct click event.
- 3 Define the other attributes based on how they map to the parameters from the API. Use [Table 28](#) for reference. When you define matching rules for these attributes, you can match on anything. For example, you can match on only one "flightbooking" rule, or you can define multiple rules that are differentiated by the airline.

This is an example of the Page Event section of a Goal that matches the click event:

Page Event

Event: Click

Target: id *	Equals	flightbooking	
Target: name *	Regular expression	.*	
Target: alt *	Regular expression	.*	
Target: selectorPath *	Regular expression	.*	
Target: tagName *	Regular expression	.*	
Target: onClick *	Regular expression	.*	
Target: innerText *	Regular expression	.*	

Example: Click Event with HTML5 Custom Data Attributes Using the same scenario as the previous example, you could make the data more user-friendly by using HTML5 custom attributes. Instead of using the built-in parameters, you could define the click event like this:

```
com_sas_ci_acs.ap.i("event", "send", {
  "event": "click",
  "data-eventtype": "flightbooking",
  "data-departure-airport": "RDU",
  "data-arrival-airport": "LHR",
  "data-airline": "SAS Airlines",
  "data-departure-date": "2019-12-25",
  "data-departure-time": "18:05",
  "data-class-of-travel": "Economy"
});
```

In the user interface, you would then define a [custom JavaScript API event](#) in **Advanced Collection Rules**. In the **Page Event** section, complete these steps:

- 1 Select **Click** from the **Event** menu.

- 2 Create an HTML custom data attribute. Set the name to **data-eventtype** and specify that the attribute's value must equal **flightbooking**.
- 3 Define the other attributes as HTML5 custom data attributes.

This is an example of the Page Event section of a Goal that matches the click event:

Page Event

Event: Click ▼

HTML5 custom data attributes:

data-eventtype	Equals ▼	flightbooking	
data-departure-airport	Regular expression ▼	.*	
data-arrival-airport	Regular expression ▼	.*	
data-airline	Regular expression ▼	.*	
data-departure-date	Regular expression ▼	.*	
data-departure-time	Regular expression ▼	.*	
data-class-of-travel	Regular expression ▼	.*	



Content Change Events

Content change events track when the content of a web page changes. Here are the parameters for this type:

Table 29 Parameters for Contentchange Events

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be contentchange .
target_name	Required	String	The name of the target that is associated with the event.
target_selector	Required	String	The selector information of the target that is associated with the event.
attribute_name	Optional	String	The name of an attribute that is associated with the event.

Name	Required or Optional	Type	Description
event_id	Optional	String	An ID for the event.
event_uid	Optional	String	A UID for the event. A UID is an ID that is globally unique across your website or app. By contrast, an event_id might be an ID that is used across multiple pages.
event_source	Optional	String	The source of an event.
new_value	Optional	String	The new value for an element or attribute.
old_value	Optional	String	The previous value for an element or attribute.
target_id	Optional	String, Number	The ID of the target that is associated with the event.
target_type	Optional	String	The type of content change. The value must be one of these strings: ■ attribute ■ childadd ■ childremove

This is an example of a content change event:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "contentchange",
  "target_selector": "div",
  "target_name": "content_div",
  "event_id": "45",
  "event_uid": "12458796"
});
```

Identity Events

Identity events track when a customer identifies him or her self (for example, by signing in). An identity event from the JavaScript Events API can replace or supplement identity events in the user interface. You do not need to perform additional configuration steps to ensure these identity events work with corresponding events in the user interface.

Here are the parameters for this type:

Table 30 *Parameters for Identity Events*

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be IdentityEvent .
login_event	Required	String	The value of the login event, such as the customer ID or login ID. Use object/value pairs if you need to obscure sensitive information. See the example after this table.
login_event_type	Required	String	The type of identity. Use one of these values: ■ customer_id ■ login_id ■ email_id Note: This value is deprecated. In most cases you should use one of the other values.

This is an example of an identity event:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "IdentityEvent",
  "login_event": "sample_email@server.com",
  "login_event_type": "login_id"
});
```

This is an example of an identity event that uses an obscured email address:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "IdentityEvent",
  "login_event": {
    "value": "sample_email@server.com",
    "obscured": true
  },
  "login_event_type": "login_id"
});
```

Impression-related Events

Impression-related events enable you to track when a creative is used on your web page. These events are used to complement scenarios in which you need more control over how content is displayed on your page and how users can interact with it.

There are three types of events that you can send with the JavaScript API:

- Impression (spot change) events track when a creative is delivered by SAS Customer Intelligence 360. Impression events do not typically take into account whether a user actually views the content or interacts with it.

- Impression viewable (spot viewable) events track when a creative is displayed in your content. How you define “viewable” can be specific to your needs. For example, you might decide that a creative is viewable when a user scrolls past its position on the page.
- Impression click (spot click) events track when a user interacts with content in a spot. You can send only 50 impression click events for each user session.

Here are the parameters for these impression event types:

Table 31 *Parameters for Impression-related Events*

Name	Required or Optional	Type	Description
event	Required	String	The event type. Set the value based on the event type: ■ For impression (spot change) events, set the value to spot_change. ■ For impression viewable (spot viewable) events, set the value to spot_viewable. ■ For impression click (spot click) events, set the value to spot_click.
spot_id	Required	String	The ID of the spot on the page.
creative_id	Required	String	The ID of the creative that was delivered, viewed, or clicked.
task_id	Required	String	The ID of the associated task.
variant_id	Optional	String	The ID of the creative's variant, if applicable.

This is an example of an impression (spot change) event:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "spot_change",
  "spot_id": "d57d4ad1-1b3b-42f6-9bf5-4b163480ed16",
  "creative_id": "48a4f271-03cd-48d4-ab08-c048b6ed5999",
  "task_id": "bb9606fd-1a8c-458d-845a-ea7d3cbf8bf2"
});
```

JavaScript Variable Change Events

JavaScript variable change (jsvarchange) events track when a JavaScript variable changes on a web page. Here are the parameters for this type:

Table 32 *Parameters for Jsvarchange Events*

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be jsvarchange .
action	Required	String	Specifies how the variable has changed. Use one of these values: ■ add ■ update The default value is update .
name	Required	String	The name of the variable that changed.
new_value	Required	String, Number	The new value of the variable.
old_value	Optional	String, Number	The previous value of the variable.

This is an example of a jsvarchange event:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "jsvarchange",
  "name": "isLoggedIn",
  "action": "update",
  "old_value": "false",
  "new_value": "true"
});
```

Logout Events

Logout events enable you to send a JavaScript call to the system that logs out the current user. This feature enables you to recategorize the user as an anonymous visitor so that creatives and messages are no longer customized with merge tags associated with the known identity.

When you send a logout event, SAS Customer Intelligence 360 performs these steps:

- 1 Disassociates the visitor ID from the datahub ID.
- 2 Requests a new datahub ID. This new datahub ID is anonymous, so it has no merge tags or uploaded data associated with it.
- 3 Associates all subsequent events with the new anonymous datahub ID.

This is an example of a logout event:

```
com_sas_ci_acs.ap.i("event", "send", {
  "event": "logout"
});
```

Logout events have no additional parameters.

Media Events

Media events track when a customer interacts with a media element. Here are the parameters for this type:

Note: For each page load, the number of media events is limited to 200.

Table 33 *Parameters for Media Events*

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be media .
media	Required	String	The name or type of media. The value must be less than or equal to 260 characters.
media_action	Required	String	The action that took place. Use one of these values: ■ start ■ stop ■ end ■ playing
media_duration	Required	Number	The length of the media. The exact format and value of this attribute depend on the values returned by the media player, but the value should be submitted as an integer or decimal.
media_position	Required	Number	The current position of the playback controller. The exact format depends on the values returned by the media player, but the value should be submitted as an integer.
player_id	Required	String	The ID of the media player's element.
media_player	Optional	String	A string or number that identifies the media player. The value must be less than or equal to 30 characters.

This is an example of a media event:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "media",
  "player_id": "154254",
  "media": "intro_video",
  "media_player": "Brightcove",
  "media_position": "314",
  "media_duration": "720",
```

```
"media_action": "start"
});
```

Submit Events

About Submit Events Submit events track when a customer submits a form. When you send a submit event with the API, the data is sent directly to the FORM_DETAILS table. You do not need to perform additional configuration steps to populate data in that table.

In addition, submit events can be used in combination with other events or advanced configurations. For example, you can use a submit event as part of a business process or goal. For an example, see [“Example: Customer Feedback Form in a Goal”](#).

Here are the parameters for this type:

Table 34 Parameters for Submit Events

Name	Required or Optional	Type	Description
event	Required	String	The event type. The value must be <code>submit</code> .
form_h_id form_h_name form_o_id	Required	String	An ID or name for the form. These parameters can be used to trigger collection rules (such as a goal or business process) that depend on the form ID or name. TIP For best results, include all three parameters and set them to the same value. If you have a goal or business process that is based on this event: - the form ID in your rule is compared with the <code>form_h_id</code> value from this event - the form name in the rule is compared with the <code>_form_h_name</code> value from this event
form_o.attempts	Required	Number	The number of attempts or submitting the form. The value must be an integer. To maintain an accurate count, increment a count variable for each attempt and use that value for this property.
Field-specific parameters	Required		One or more parameters for a field. Note: Submit events must include parameters for at least one form field. See Table 35 for these parameters.

To send information about the fields in a form, you must include a set of parameters for each field. For each field, set these parameters:

Table 35 Field Parameters for Submit Events

Name	Required or Optional	Type	Description
<code>form.f.<field_name>._h.name</code>	Required	String	The name or ID of a form field.
<code>form.f.<field_name>._o.changes</code>	Required	Number	The number of times (as an integer) that a form field is changed before the form is submitted. To maintain an accurate count, increment a count variable for each attempt and use that value for this property. TIP In cases where you cannot determine this value, it can be set to 1.
<code>form.f.<field_name></code>	Optional	String	The value of a form field. Use object/value pairs if you need to obscure sensitive information. For example, this name/value pair defines the value for the <code>userId</code> field: <pre>"form.f.userId": "myUserID"</pre>
<code>form.f.<field_name>._h.type</code>	Optional	String	The type of value expected for a form field.

TIP When you construct the field-specific form parameters, ensure that there are no extra spaces between the parameter name and the quotation marks. Extra spaces can cause the field value to be ignored.

This is an example of a submit event:

```
com_sas_ci_acs.ap.i("event", "send", {
  "event": "submit",
  "form._o.id": "simple_form",
  "form_h_id": "simple_form",
  "form_h_name": "simple_form",
  "form._o.attempts": 1,
  "form.f.Q1FavoriteColor._h.name": "Q1FavoriteColor",
  "form.f.Q1FavoriteColor": $scope.Q1,
  "form.f.Q1FavoriteColor._o.changes": 1,
});
```

Example: Customer Feedback Form in a Goal In this example, you want to send a submit event that contains data from a customer feedback form. You also want to define a [custom JavaScript API event](#) to record when this form is submitted.

For each field in the form, you must send this information:

- the field's name
- the field's value

- the number of changes made to the field. In the example, this value is set to 1 for simplicity.

This is an example of a submit event that captures this information from a customer feedback form with seven questions:

```
com_sas_ci_acs.ap.i("event", "send",
{
  "event": "submit",
  "form._o.id": "feedback_form",
  "form_h_id": "feedback_form",
  "form_h_name": "feedback_form",
  "form._o.attempts": 1,
  "form.f.Q1HowSatisfiedWereYou._h.name": "Q1HowSatisfiedWereYou",
  "form.f.Q1HowSatisfiedWereYou": $scope.Q1,
  "form.f.Q1HowSatisfiedWereYou._o.changes": 1,
  "form.f.Q1a._h.name": "Q1a",
  "form.f.Q1a": $scope.Q1a,
  "form.f.Q1a._o.changes": 1,
  "form.f.Q2._h.name": "Q2",
  "form.f.Q2": $scope.Q2,
  "form.f.Q2._o.changes": 1,
  "form.f.Q2a._h.name": "Q2a",
  "form.f.Q2a": $scope.Q2a,
  "form.f.Q2a._o.changes": 1,
  "form.f.Q3._h.name": "Q3",
  "form.f.Q3": $scope.Q3,
  "form.f.Q3._o.changes": 1,
  "form.f.Q4._h.name": "Q4",
  "form.f.Q4": $scope.ratedBy,
  "form.f.Q4._o.changes": 1,
  "form.f.Q5._h.name": "Q5",
  "form.f.Q5": $scope.Q5,
  "form.f.Q5._o.changes": 1,
  "form.f.Q6._h.name": "Q6", /* if this field is an email,
                             you might want to obscure it. */
  "form.f.Q6": $scope.Q6,
  "form.f.Q6._o.changes": 1,
  "form.f.Q7._h.name": "Q7",
  "form.f.Q7": $scope.country,
  "form.f.Q7._o.changes": 1
});
```

In the user interface, you would then define a [custom JavaScript API event](#) in **Advanced Collection Rules**. In the **Page Event** section, complete these steps:

- 1 Select **Form Submit** from the **Event** menu.
- 2 Add the Form:id attribute and specify that the attribute's value must equal `feedback_form`.

This is an example of the Goal's Page Event section:

Page Event

Event:

Form: id *

When you send the submit event with the JavaScript Events API, the Goal in the user interface captures this event and is recorded.

Server-Side Events API

About the Server-Side Events API

IMPORTANT The developer documentation is available at [Server-Side Events API](#).

The Server-Side Events API enables you to send web and mobile events to SAS Customer Intelligence 360 without using the [SAS tag](#) (through the JavaScript snippet) or the [SAS Mobile SDK](#). This REST API can be called from your eCommerce system, CMS, or other back-end systems.

Here is some additional information about this REST API:

- You can use the Server-Side Events API to send almost all web and mobile events through REST API calls. Server-side events are processed and stored in Unified Data Model (UDM) tables the same as events that are collected through client-side implementations of the SAS tag, JavaScript Events API, or mobile SDK.

For more information about the events that are supported, see [“Event Types” on page 79](#).

IMPORTANT To ensure data integrity in tables for the Unified Data Model (UDM), SAS Customer Intelligence 360 maintains the session IDs. Session IDs are generated automatically and comply with the session timeouts that are defined in the user interface. To see session timeouts, navigate to  **Administration** ⇒ **Tasks**. Click **Page and Session** in Digital.

- You can send events for both anonymous and identified users. For more information, see [“Send Events with the Server-Side Events API” on page 73](#).
- You can combine this REST API with event collection from the SAS tag, JavaScript Events API, or the mobile SDK. For more information, see [“Send Server-Side Events to Augment Client-Side Sessions” on page 78](#).

Prerequisites for the Server-Side Events API

If you do not already have a static JSON web token (JWT) or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an [event access point](#).

TIP When you create the access point, make note of the tenant ID. Use this value as the external tenant ID when you make calls to the REST API.

- 2 Create an authorization token. You can either [generate a static JWT](#) or use an [access token](#).
- 3 Make note of the value of the token. When you make calls to the REST API, this token is used in the Authorization header as a bearer token.

Important Considerations for Sending Events

IMPORTANT Make sure that the event's definition in the user interface is published before you send the corresponding event through the REST API.

Load Events and Sessions

In SAS Customer Intelligence 360, each session must start with a load event, and every event needs to be assigned to a page. However, SAS Customer Intelligence 360 can manage these requirements for you.

Sessions are managed automatically based on the time-out values for the channel type (web or mobile). During a session, SAS Customer Intelligence 360 can use the uri parameter from events to generate synthetic load events as needed. These load events ensure the integrity of the data (that is, every event happens on a page and every session starts with a page view).

Each time the URI changes, SAS Customer Intelligence 360 generates a new synthetic load event. For example:

- If you send multiple events with the same URI, SAS Customer Intelligence 360 generates one load event. The subsequent events are associated with that same load event.
- If you send three events, and each event contains a different URI, SAS Customer Intelligence 360 generates synthetic load events for each event.

If you are interested in recording every page load, you can send [load events on page 105](#) explicitly. If you are interested in specific events only (such as product views, purchases, and custom events), this is not necessary and you can let SAS Customer Intelligence 360 manage the load events for you.

Timing and the Order of Events

When you send events, follow these guidelines:

- Ensure that events are sent in chronological order. When events are received out of chronological order, the events are processed and assigned to the appropriate session and page load retrospectively for up to five minutes.
- In the payload for each event, you can provide a timestamp by setting the `clientTime` attribute or `clientTimeStamp` attribute. If both `clientTime` and `clientTimeStamp` are provided, SAS Customer Intelligence 360 uses the value from the `clientTime` attribute.

If neither `clientTime` nor `clientTimeStamp` are provided, SAS Customer Intelligence 360 uses the server's timestamp of when the event is received.

- If an event arrives more than five minutes after the most recent timestamp, the event is not processed. Instead, the event is stored in the [EVENT_ERRORS](#) table.
- If the timestamp in an event's payload is more or less than five minutes from the server time when the event is received, the event is stored in the [EVENT_ERRORS](#) table.

TIP If you send events individually, in real time, it might be easier to not include a timestamp in the payload. When you batch events and send multiple events in a single request, you should include the timestamp on the event to preserve the correct order. However, batched events must still arrive within five minutes of the timestamp of the request's payload.

Channels

All events are assigned to a given channel. However, you can send events for both the web channel and mobile channel for the same ID in the same request. These events are then combined in the same session.

Here is some more information about channel-specific behavior:

- Some events are available only for the web channel and not for the mobile channel. For more information, see [“Event Types” on page 79](#).
- The channel behavior for a session is based on the first event that is received for the user. For example, if a session starts with a mobile event, the session-timeout settings for the mobile channel are used. This behavior happens regardless of whether the session contains events from both channels.
- Some event types must be defined in the user interface first with an API key (for web) or Mobile App Action (for mobile). When you send these events through the REST API, make sure that the channel type corresponds to the API key that is used. If the channel is not correct, the events are discarded.

More information about specifying the key is available in these topics:

- [cart on page 84](#)
- [cart action on page 89](#)
- [custom on page 96](#)

- ❑ [internal search on page 103](#)
- ❑ [product view on page 108](#)
- ❑ [promotion on page 111](#)

How Global Settings Are Applied

Because the Server-Side Events API relies on the authentication, SAS Customer Intelligence 360 assumes that events and their corresponding values are sent from authorized sources. As a result, you are responsible for making sure that you send only events that should be processed in your environment.

Review the sections below for more information about how events from the Server-Side Events API interact with other settings in SAS Customer Intelligence 360.

Approved Domain Rules Are Not Applied

In SAS Customer Intelligence 360, you explicitly approve domains that you want to record events from. When a site includes the SAS tag, events are recorded only when the site's domain is approved and active. Otherwise, the events are rejected. To view or modify the list of approved domains, navigate to  **Administration** ⇒ **Tasks**. Click **Domains** in Digital.

The list of approved domains, and whether the domain is active, applies only to sites that include the SAS tag. These controls do not apply to events that are sent through the Server-Side Events API. In addition, this exemption also extends to data exclusion rules for approved domains. These rules are applied only when a site includes the SAS tag and not to events that are sent through the Server-Side Events API.

Mobile Application IDs Are Validated for Some Events

In the user interface for SAS Customer Intelligence 360, the list of mobile application IDs specifies which mobile apps are enabled for data collection. To view these IDs, navigate to  **Administration** ⇒ **Tasks**. Click **Mobile Applications** in Digital.

For the following event types, the mobile app ID can be set to any value and is not validated:

- [click through on page 94](#)
- [impression viewable on page 101](#)
- [load on page 105](#)

For all other event types, the events are accepted only when the mobile app ID is defined and active.

Event Type Rules Are Not Applied

In the user interface for SAS Customer Intelligence 360, you can control which event types are enabled for data collection. To view or modify these settings, navigate to  **Administration** ⇒ **Tasks**. Click **Event Tracking** in Digital.

Note: These controls apply only to events that are collected through the SAS tag.

Data Exclusion Rules Are Applied

Data exclusion rules that are related to IP address, user agents, and geo locations are defined in the administration settings. To view or modify these settings, navigate to  **Administration** ⇒ **Tasks**. Click **Event Tracking** in Digital.

If you send events through the Server-Side Events API with this information, these rules are applied. Sessions might be dropped if you send events that are excluded by these rules.

Form Field Rules Are Not Applied

SAS Customer Intelligence 360 enables you to control which forms and form fields are monitored or excluded from collection. To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Form Fields** in Digital. You can also use this page to define which form fields should be obfuscated (hashed) to prevent SAS Customer Intelligence 360 from storing potentially sensitive data.

These form controls apply only to events that are collected through the SAS tag or sent through the JavaScript Events API. These rules are not applied to events sent through the Server-Side Events API.

IMPORTANT You are responsible for obfuscating or encrypting sensitive values to prevent values from being stored in cleartext.

Interactions with User-Defined Events and Standard Events

User-Defined Events

In client-side implementations, SAS Customer Intelligence 360 uses standard events (such as load events, content-change events, and click events) as the trigger for user-defined events. In most use cases, you might use a standard event as the basis for a custom event, an add-to-cart event, or a step in a business process.

Keep these guidelines in mind when you use the Server-Side Events API with user-defined events:

- Most user-defined events can be sent through the REST API. You can send standard events as server-side events, but it is simpler to send the user-defined event directly.
- Business processes cannot be sent as server-side events, so you must trigger these events with an underlying event. For example, you must send an event such as a page load, click, or form submit to trigger the business process in SAS Customer Intelligence 360.

TIP When you define events in the user interface, make sure that have a clear distinction for attributes in events that are going to be sent only through the Server-Side Events API.

Standard Events and Data Collection

Device Lookup SAS Customer Intelligence 360 uses the client's user-agent string or the user-agent's client hints from the request header to populate information about a user's device, platform and browser.

To use this capability with the Server-Side Events API, you must send the client's user agent string or user agent's client hints in the first event of a session. As a best practice, you could send this information with every event. For more information, see [“Common Attributes” on page 79](#).

Document Downloads SAS Customer Intelligence 360 enables you to derive document download events from click events. To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Document Downloads** in Digital.

When you send events through the Server-Side Events API, you can use two options:

- You can send an explicit [document event on page 98](#).
- You can send a [click event on page 92](#), and the system derives a document event.

With either option, SAS Customer Intelligence 360 applies all the rules that are defined in the Document Downloads page.

Geo IP Lookup SAS Customer Intelligence 360 uses the client's IP address to populate information about a user's geographic location (such as country, state or region, city, and so on). To use this capability with server-side events, you must send the client's IP address in the payload of the first event of a session. As a best practice, you can also send this information with every subsequent event. For more information, see [“Common Attributes” on page 79](#).

To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Personal Identifiable Information** in Digital.

For all events that are sent through the Server-Side Events API, SAS Customer Intelligence 360 applies these settings, the same way the settings would be applied to events from the SAS tag or mobile SDK.

Forms SAS Customer Intelligence 360 enables you to rename forms when data is collected. The feature is typically used when the form's name is dynamically generated or hard to decipher. To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Forms** in Digital.

For any submit events that are sent through the Server-Side Events API, SAS Customer Intelligence 360 applies the form rules.

When you send a submit event, you must set the name of the form in the event's payload. As a best practice, send form names that are human readable and meaningful in your data.

Page Categories SAS Customer Intelligence 360 enables you to define rules that assign records to up to 15 different [page categories](#). These page categories could describe a content hierarchy or other page attributes like the page's language. To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Page Categories** in Digital.

For all events that you send through the Server-Side Events API, SAS Customer Intelligence 360 applies the page category rules the same way they would be applied to events from the SAS tag or the mobile SDK.

Note: If your page categories depend on event attributes, make sure that you send these attributes with your server-side events.

Page URLs SAS Customer Intelligence 360 enables you to define rules to modify the page URI that is stored in the PAGE_DETAILS table. These rules are commonly used to drop irrelevant URL query parameters (like jsessionid, which could create a large number of distinct URLs). To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Page URLs** in Digital.

For any load events that are sent through the Server-Side Events API, SAS Customer Intelligence 360 applies the rules for page URLs.

When you send events through the Server-Side Events API, you control the URI value when you send events that contain the page URI. Format the URI to include only the relevant query parameters, and remove any parameters that are irrelevant or might contain sensitive data.

Page Views SAS Customer Intelligence 360 enables you to generate synthetic page views from click, content change, field interaction, form submit or JavaScript variable change events. These synthetic page view events are typically used to record page views for Single Page Applications (SPAs). To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Page Views** in Digital.

When you send events through the Server-Side Events API, you should send explicit [load events on page 105](#) for any content changes in an SPA.

In addition, for any click and submit events that you send through the Server-Side Events API, SAS Customer Intelligence 360 applies the rules for page views.

Traffic Sources SAS Customer Intelligence 360 enables you to define rules that determine the source of incoming traffic. Typically, these rules are used to parse the UTM query parameters in the URLs of landing pages. To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Traffic Sources** in Digital.

For all events that you send through the Server-Side Events API, SAS Customer Intelligence 360 applies the traffic source rules the same way they would be applied to events from the SAS tag or the mobile SDK. Make sure that send the attributes that are used in your traffic source definitions.

Visitor Profiles SAS Customer Intelligence 360 enables you to derive visitor profiles from attributes of the landing page. These rules are typically used to classify sessions from internal employees (based on IP address, user agent, or geo location information), so those sessions are excluded from customer analysis. To view or modify these rules, navigate to  **Administration** ⇒ **Tasks**. Click **Visitor Profiles** in Digital.

For load events that you send through the Server-Side Events API, SAS Customer Intelligence 360 applies the rules for visitor profiles. Make sure that you send any attributes with the event that are used for visitor profiles.

Send Events with the Server-Side Events API

How to Send Events

Use the information in this topic when you are sending events based on your use case:

- [“Send Events for Identified \(Known\) Users” on page 75](#)
- [“Send Events for Anonymous Users” on page 77](#)
- [“Send Server-Side Events to Augment Client-Side Sessions” on page 78](#)

IMPORTANT You are responsible for the events and values that you send through the Server-Side Events API. SAS does not filter traffic or obfuscate any values that are sent through this REST API.

Use the following steps as a guideline for how to submit events:

- 1 Set the URL for a server-side event based on this basic example:

POST https://<server-side host name>/...

Use one of the following host names, based on your tenant's region:

- United States: **i-us.ci360.marketing**
- Europe: **i-eu.ci360.marketing**
- Asia Pacific Northeast: **i-ap.ci360.marketing**
- Sydney: **i-au.ci360.marketing**
- Mumbai: **i-mu.ci360.marketing**

The remainder of the URL is based on whether the event is for known users or anonymous users.

2 For the header, set these values:

- a Set the Authorization header to use a Bearer token that is set to static JWT or an access token. For more information, see [“Prerequisites for the Server-Side Events API” on page 68](#). This authorization token helps prevent traffic spoofing from third parties.

As best practices, follow these guidelines:

- Do not make calls to the Server-Side Events API directly from within a client's browser, because these calls would expose the value of the authorization token. When you need to send events from the client's browser, you should use the [JavaScript Events API on page 3](#) instead.
- When you make calls from a mobile app, ensure that the authorization token is properly protected and cannot be accessed by the user.

- b Set the content-type attribute to **application/x-ndjson**.

3 In the body of the request, add the JSON payload with the event's information. For information about the supported events, see [“Event Types” on page 79](#).

The payload must use the newline-delimited JSON (x-ndjson) format, where rows are delimited by a newline character (\n). You can send up to 15 rows with each request, and each row represents a single event.

Note: Each event must be sent as a single row. If you send an event in multi-line format, the REST call fails.

Here are examples of event payloads that could be in the body of a request:

```
{ "eventName": "load", "channel": "web", "clientTime": "2025-11-14T15:45:02.345Z", "uri": "https://www.sas.com/books.html", "properties": { "page_title": "SASBooks", "referrer": "https://www.google.com" } }
```

```
{ "eventName": "productView", "channel": "web", "clientTime": "2025-11-14T15:45:05Z", "uri": "https://www.example.com/products", "apiEventKey": "view_product", "product": { "id": "PROD-001", "name": "WidgetPro", "unitPrice": 99.99 } }
```

```
{ "eventName": "cartAction", "apiEventKey": "addBook", "channel": "web", "clientTime": "2025-11-14T15:45:04.802Z", "uri": "https://www.sas.com/books.html", "product": { "sku": "978-1-394-25872-7", "unitPrice": 60 } }
```

Send Events for Identified (Known) Users

When you initiate a session from the Server-Side Events API for a known user, you can send events using either the visitor ID or a known ID.

- To send events with only the visitor ID, first send an [attach-identity event on page 100](#) to associate the visitor ID with an ID type in SAS Customer Intelligence 360 (such as customer_id, subject_id, or login_id value). You can then send any subsequent events using only the visitor ID, and these events are associated with the known ID.

When you send events with a visitor ID, SAS Customer Intelligence 360 looks up if there is a known identity (subject_id, customer_id or login_id) that is associated with that visitor_id value. This function works the same as it does through the SAS tag or the mobile SDK.

- To send events with a known ID and value, you can use these values in the URL as query parameters. For more information, see [“Send Events Using a Known ID” on page 75](#).

IMPORTANT If the session is already established by the SAS tag or mobile SDK, and you are sending events to augment these sessions, you do not need to attach an identity or use an ID value. For more information, see [“Send Server-Side Events to Augment Client-Side Sessions” on page 78](#).

Send Events Using a Known ID

When you already have the identity type and value for a user, you can send events to SAS Customer Intelligence 360 without attaching an identity first. In the request's URL, you include these values as query parameters in the URL.

- 1 To send an event with a known ID type, send a POST call to this endpoint:

```
POST https://<server-side host name>/t/events/e/<external tenant ID>/
id_type=<identity_type>/id_value=<identity_value>
```

This table describes how to set these query parameters:

URL Parameter	Description
id_type	Identity type. Use one of these values: ■ subject_id ■ login_id ■ customer_id
id_value	Corresponding value of the identity type

- 2 Send the event based on the information in [“How to Send Events” on page 73](#). For information about the response, see [“Responses, Errors, and Troubleshooting” on page 78](#).

Note: If you are augmenting the mobile SDK with server-side events for a known identity, and you want to use a server-side event to trigger an in-app message, you must pass the application's device ID as the visitor ID.

- 1 To retrieve the device ID, use one of these methods in the SDK:

- For Android, use `SASCollector.getInstance().getDeviceID()`;
- For iOS, use `SASCollector.device()`

- 2 Send the triggering event with a POST request like this example:

```
POST https://<server-side host name>/t/events/e/<externalTenantId>/<application device ID>
```

Send Events for an Attached Identity

Attach-identity requests associate a visitor ID with a known ID in SAS Customer Intelligence 360. Complete these steps to send an attach-identity request:

- 1 Determine the visitor ID. You can use a visitor ID that is already available in your system, or you can request a visitor ID from SAS Customer Intelligence 360.

- **Using your own visitor ID:** make sure that your visitor ID complies with these rules:
 - ☐ is a hexadecimal value that contains only numbers 0 to 9, the letters A through F, or the letters a through f
 - ☐ is a minimum of eight characters long
 - ☐ is a maximum of 128 characters long
 - ☐ follows case-sensitive rules

- **Requesting a visitor ID:**

- 1 Send this call to request the visitor ID:

```
GET https://i-<region>.ci360.marketing/t/events/i/<external tenant ID>
```

The JSON response is similar to this example:

```
{ "visitorId": "<value>" }
```

- 2 Send an [attach-identity event on page 100](#) to associate the visitor ID with the known IDs in SAS Customer Intelligence 360.
- 3 Send an event based on this information:
 - In the URL for the REST call, include the visitor ID from the previous step:


```
POST https://<server-side host name>/t/events/e/<external tenant ID>/<visitor ID>
```
 - Send the event based on the information in [“How to Send Events” on page 73](#). For information about the response, see [“Responses, Errors, and Troubleshooting” on page 78](#).
- 4 (Optional) To disconnect a known identity ID from a visitor ID, send a [detach-identity event on page 100](#).

See Also

[“Send Events for Anonymous Users” on page 77](#)

Send Events for Anonymous Users

When the session is purely server side and uses only the Server-Side Events API (and optionally the Content Request API), send events for anonymous users by using a visitor ID in the URL.

Note: When the session also uses the SAS tag (either the full tag or the JavaScript Events API) or the mobile SDK, see [“Send Server-Side Events to Augment Client-Side Sessions” on page 78](#).

Complete these steps:

- 1 Determine the visitor ID. You can use a visitor ID that is already available in your system, or you can request a visitor ID from SAS Customer Intelligence 360.

- **Using your own visitor ID:** make sure that your visitor ID complies with these rules:

- ☐ is a hexadecimal value that contains only numbers 0 to 9, the letters A through F, or the letters a through f
- ☐ is a minimum of eight characters long
- ☐ is a maximum of 128 characters long
- ☐ follows case-sensitive rules

- **Requesting a visitor ID:**

- 1 Send this call to request the visitor ID:

```
GET https://i-<region>.ci360.marketing/t/events/i/<external tenant ID>
```

The JSON response is similar to this example:

```
{ "visitorId": "<value>" }
```

- 2 Send the event based on the information in [“How to Send Events” on page 73](#), but use this additional information:

- If you are sending a spot-related event, make sure that you first make a content request with the same visitor ID through the [Content Server Request API on page 126](#).
- Send a REST call to the endpoint URL that includes the visitor ID:

```
POST https://<server-side host name>/t/events/e/<external tenant ID>/<visitor ID>
```

IMPORTANT Use the same visitor ID for all events that are associated with a specific device or browser.

For information about the response, see [“Responses, Errors, and Troubleshooting” on page 78](#).

- 3 (Optional) If the visitor becomes a known user during the session, send an [attach-identity event on page 100](#) to associate the visitor ID with known IDs. After you attach the identity, you can continue using the visitor ID when you send server-side events.

See Also

[“Send Events for Identified \(Known\) Users” on page 75](#)

Send Server-Side Events to Augment Client-Side Sessions

About Augmenting Sessions

When your site or mobile app uses the SAS tag or the mobile SDK, you can use server-side APIs to augment those sessions.

For example, consider these scenarios:

- You might need to capture relevant information that is not available or accessible to the SAS tag or mobile SDK in the client. You can then send the event with all the details through the Server-Side Events API (for example, cart events or cart-action events).
- You can use the Server-Side Events API to attach an identity to a visitor when they log in. Some sites store only an access token on the client side, but the known ID (subject_id, customer_id or login_id) is not available on the client. This configuration prevents you from attaching the identity with the JavaScript Events API. In this situation, you send an [attach-identity event on page 100](#) through the Server-Side Events API.

Sending Events for Augmented Sessions

- 1 Send the event data as a POST call like this example:

```
POST https://<server-side host name>/t/events/e/<external tenant ID>/<visitor ID>
```

- 2 In the body of the request, include the session ID and load ID in the event's payload to ensure that the events are associated with the correct session, visitor, identity, and page. To get these values, you can use one of these methods:
 - For web sessions, you can use the SAS tag to retrieve the IDs. Enter the following commands in the browser's console:
 - For the session ID: `ci360("info").sid`
 - For the load ID: `ci360("info").loadId`
 - In the mobile SDK, you can use these methods to get the session ID and load ID (respectively):
 - For Android, use these methods:


```
SASCollector.getInstance().getSessionID();
SASCollector.getInstance().getLoadID();
```
 - For iOS, use these methods:


```
SASCollector.getSessionID();
SASCollector.getLoadID();
```
- 3 Send the event based on the information in [“How to Send Events” on page 73](#). For information about the response, see [“Responses, Errors, and Troubleshooting” on page 78](#).

Responses, Errors, and Troubleshooting

Responses and Errors

When you send the event, the POST request can return these statuses:

- 4xx errors are returned when the URL is invalid. The URL can be invalid when the external tenant ID in the URL is missing or invalid, or when the structure of the URL is incorrect.
- The 200 (OK) status is returned in all other cases.

There are no immediate checks for the validity of the data in the event's payload (such as invalid event names or missing fields). Errors and inconsistencies that cause events to be discarded can be reviewed in the [EVENT_ERRORS](#) table.

Troubleshooting

If you want to troubleshoot or monitor errors:

- 1 Create a [general access point](#). During the [step when you define associations](#), add the Error event.
 - a Click **+**
 - b Type "error" in the search field and select **Error**.

Finish the remaining configuration steps and click **Done**.
- 2 Install a general agent on premises that is configured for the access point.
- 3 Monitor the agent for errors that might stream in.

Event Types

Common Attributes

Common Attributes for Events

For all events, the following common attributes are required:

- eventName
- channel
- uri

TIP In addition, as a best practice, you should include either the clientTime attribute or the clientTimeStamp attribute on every event. For more information about how these attributes are used, see ["Timing and the Order of Events" on page 69](#).

The following table describes the common attributes for server-side events:

Table 36 Common Attributes for Events

Object	Attribute	Required or Optional	Data Type	Description
root	eventName	Required	String	Name of the event that is being sent. You must use one of these values: ■ load ■ click ■ submit ■ media ■ cart ■ cartAction ■ custom ■ internalSearch ■ productView ■ promotion ■ document ■ spot_viewable ■ spot_clicked
root	channel	Required	String	Channel for the event. You must use one of these values: ■ web: refers to a browser-based website. This value can be used regardless of whether the site is accessed from a desktop PC or a mobile device. ■ mobile: refers to a mobile app.
root	uri	Required	String	Full URI of the page that is associated with the event. The URI should include the protocol, host name, path, and any anchors or query parameters that are necessary to identify the page. IMPORTANT The uri attribute from the first event can be used to generate a synthetic page-load event. For more information, see “Load Events and Sessions” on page 68. To store the URI correctly, some characters must be HTML encoded. These are the only characters that can be unencoded: ■ alphanumeric characters ■ special characters: \$-_.+!*'(), ■ reserved characters: ;/?:@=&

Object	Attribute	Required or Optional	Data Type	Description
				For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt .
root	apiEventKey	Required for non-system events	String	Event key from the event's definition in the user interface. This value is required for non-system events. ■ For the mobile channel, set the key in the user interface in Events ⇒ <Event Type> ⇒ Mobile App Action. When you send the event, the event's payload must include the mobile event key from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event. ■ For the web channel, set the key in the user interface in Events ⇒ <Event Type> ⇒ Event API. When you send the event, the event's payload must include the API event key from this definition. IMPORTANT Make sure that apiEventKey value and the channel type correspond to the API key for the correct event type in the user interface. If the channel is not correct, events are discarded.
root	clientTime	Optional	String	UTC timestamp when the event happened. The timestamp must use the ISO8601 format, like 2025-11-11T03:03:07.385Z . For more information, see “Timing and the Order of Events” on page 69 .
root	clientTimeStamp	Optional	String	UTC timestamp when the event happened. The timestamp must use the UNIX epoch format in milliseconds, like 1772040885 . For more information, see “Timing and the Order of Events” on page 69 .
root	sessionId	Optional	String	Session ID that is generated by the SAS tag or the mobile SDK. You can send a sessionId attribute to add a server-side event to an existing client-side session (for example, you could send a cart event directly from an eCommerce back-end system). For more information, see “Send Server-Side Events to Augment Client-Side Sessions” on page 78. Note: When you include a sessionId attribute, you must also set the loadId attribute.

Object	Attribute	Required or Optional	Data Type	Description
root	loadId	Optional	String	Load ID that is generated by the SAS tag or the SAS mobile SDK.

Common Session Attributes

The following attributes set values that are used to populate the [SESSION_DETAILS](#) table.

IMPORTANT These attributes must be sent on the first event for the session. For the web and mobile sections, you can also send the attributes that are listed as optional with subsequent events for that channel.

Table 37 Common Session Attributes

Object	Attribute	Required or Optional	Data Type	Description
properties	ip	Optional	String	IP address of the client. When this attribute is provided, it is used for the geo IP lookup based on the administration settings. To view this page, navigate to  Administration ⇒ Tasks. Click Personally Identifiable Information in Digital. These settings then populate the values for the attributes country_nm, country_cd, state_region_cd, region_nm, city_name, latitude, longitude, organization_nm, postal_cd and metro_cd.
mobile	appld	Required for mobile events	String	App ID that is used to configure mobile events, which is defined in the administration settings. To view this page, navigate to  Administration ⇒ Tasks. Click Mobile Applications in Digital. For the following event types, the mobile app ID can be set to any value and is not validated: ■ click through on page 94 ■ impression viewable on page 101 ■ load on page 105

Object	Attribute	Required or Optional	Data Type	Description
mobile	language	Optional (and specific to mobile events)	String	Language setting for your app or the device.
mobile	appVersion	Optional (and specific to mobile events)	String	Version of your mobile app.
mobile	countryCode	Optional (and specific to mobile events)	String	Three-digit mobile country code (MCC).
mobile	networkCode	Optional (and specific to mobile events)	String	Two-digit or three-digit mobile network code (MNC)
mobile	carrierName	Optional (and specific to mobile events)	String	Name of the mobile carrier.
mobile	screenHeight	Optional (and specific to mobile events)	String	Height of the mobile device's screen, in pixels.
mobile	screenWidth	Optional (and specific to mobile events)	String	Width of the mobile device's screen, in pixels.
mobile	deviceModel	Optional (and specific to mobile events)	String	Name of the mobile device. For example, the value might be iPhone 16 Plus or SM-S921B. The deviceMfg value plus this field's value populates the value for the device_nm attribute. If the deviceMfg value is not provided, the default value of Make is used.
mobile	deviceType	Optional (and specific to mobile events)	String	Type of the mobile device. For example, the value might be phone or tablet .
mobile	deviceMfg	Optional (and specific to mobile events)	String	Manufacturer of the mobile device. For example, the value might be Samsung or Apple. The deviceModel value plus this field's value populates the value for the device_nm attribute. If the deviceModel value is not provided, the default value of Model is used.

Object	Attribute	Required or Optional	Data Type	Description
mobile	platform	Optional (and specific to mobile events)	String	Platform for the mobile device. For example, the value might be <code>iOS</code> or <code>Android</code> .
mobile	platformVersion	Optional (and specific to mobile events)	String	Version of the mobile device's platform. For example, the value might be <code>26.1</code> or <code>16</code> .
mobile	deviceLanguage	Optional (and specific to mobile events)	String	Language that is set for the mobile device. For example, the value might be <code>en</code> or <code>de</code> .
web	userAgent	Optional (and specific to web events)	String	User-agent string from the client's browser. This string can be used to determine information about the device and platform.
web	sec-ch-ua*	Optional (and specific to web events)	String	User-agent hints from the client's browser. This string can be used to determine information that is related to the device or platform. Note: You must escape quotation marks within the value with a backslash. For example: <pre>"sec-ch-ua": "Chromium;v=\"122\"", \"Not (A:Brand\";v=\"24\"\", \"Google Chrome \";v=\"122\""</pre>

Cart Event

Cart events record information about the current content of a shopping cart at the time the user is viewing it. SAS Customer Intelligence 360 distinguishes between three different stages of cart events:

- 1 A *cartview* indicates that the user viewed the contents of their cart.
- 2 A *checkout* indicates that the user started the purchase process. This event includes the items that are in the cart before the purchasing (or before the purchase process is abandoned).
- 3 Finally, *purchase* indicates that the customer completed the purchase process. This record includes the items that were purchased.

Before you send this type of event, you must define these items in the user interface:

- The key for the event.
 - For the mobile channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Mobile App Action**. When you send the event, the event's payload must include the mobile event key

from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event.

- For the web channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Event API**. When you send the event, the event's payload must include the API event key from this definition.
- (Optional) Custom attributes. When you want to send custom attributes with an event, these attributes must exist in the respective cart-related event in the user interface.

These definitions enable the attributes to be used in SAS Customer Intelligence 360 and to be stored in the `properties_map_doc` field of the [ORDER_SUMMARY](#) table.

Note: Custom attributes are available only at the cart level and not on the cart items.

This table describes the attributes for this event type:

Table 38 *Attributes for Cart Events*

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be <code>cart</code> .
cart	action	Required	String	ORDER_SUMMARY.RECORD_TYPE	Stage of the cart in the checkout process. You must use one of these values: ■ <code>cartview</code> ■ <code>checkout</code> ■ <code>purchase</code>
cart	id	Required	String	ORDER_SUMMARY.CART_ID	ID of the cart. You must set either the <code>id</code> attribute or the <code>name</code> attribute.
cart	name	Required	String	For informational purposes only	Name of the cart. You must set either the <code>id</code> attribute or the <code>name</code> attribute.
cart	billingCity	Optional	String	ORDER_SUMMARY.BILLING_CITY_NM	City where the invoice should be sent. The maximum length for this value is 390 characters.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
cart	billingCountry	Optional	String	ORDER_SUMMARY.BILLING_COUNTRY_NM	Country where the invoice should be sent. The maximum length for this value is 85 characters.
cart	billingPostcode	Optional	String	ORDER_SUMMARY.BILLING_POSTAL_CD	Postal code where the invoice should be sent. The maximum length for this value is 10 characters. For example, in the United States this value could be the ZIP code.
cart	billingRegion	Optional	String	ORDER_SUMMARY.BILLING_STATE_REGION_CD	Region where the invoice should be sent. The maximum length for this value is 256 characters. For example, in the United States this value could be the state.
cart	shippingCity	Optional	String	ORDER_SUMMARY.SHIPPING_CITY_NM	City where the items should be sent. The maximum length for this value is 390 characters.
cart	shippingCountry	Optional	String	ORDER_SUMMARY.SHIPPING_COUNTRY_NM	Country where the items should be sent. The maximum length for this value is 85 characters.
cart	shippingPostcode	Optional	String	ORDER_SUMMARY.SHIPPING_POSTAL_CD	Postal code where the items should be sent. The maximum length for this value is 10 characters. For example, in the United States this value could be the ZIP code.
cart	shippingRegion	Optional	String	ORDER_SUMMARY.SHIPPING_STATE_REGION_CD	Region where the items should be sent. The maximum length for this value is 256 characters. For example, in the United States this value could be the state.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
cart	deliveryType	Optional	String	ORDER_SUMMARY.DELIVERY_TYPE_DESC	Delivery type. The maximum length for this value is 42 characters. For example, the value might be standard or Next Day .
cart	items	Optional	Object	Members of this object are written to the ORDER_DETAILS table.	An object that defines items in the cart by name/value pairs. Each row uses the same attributes as the product section of the cart-action event on page 89 . These are the attributes that you can include in this array: ■ id (String) ■ name (String) ■ sku (String) ■ group (String) ■ quantity (Number) ■ unitPrice (Number) ■ currencyCd (String) ■ availabilityMessage (String) ■ savingsMessage (String) ■ shippingMessage (String)
cart	orderId	Optional	String	ORDER_SUMMARY.ORDER_ID	Customer's unique order ID, which should be generated after a customer completes an order. Ensure that order IDs are unique for each customer's order. Order IDs can be used to link other back-end data that is related to the order (such as returns or cancellations).
cart	paymentType	Optional	String	ORDER_SUMMARY.PAYMENT_TYPE_DESC	Payment type. The maximum length for this value is 42 characters. For example, the value might be ApplePay , PayPal , Credit , and so on.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
cart	tax	Optional	Number	ORDER_SUMMARY.TOTAL_TAX_AMT	Tax amount. The value must be an integer or a decimal.
cart	totalCartValue	Optional	Number	ORDER_SUMMARY.TOTAL_PRICE_AMT	Total value for items in the cart. The value must be an integer or a decimal. Decide whether this value includes taxes and shipping costs, and be consistent for every order.
properties	<custom property name>	Optional			Custom properties that are associated with the event. Make sure that the custom properties are defined for the event in the user interface.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a cart event:

```
{
  "eventName": "cart",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:05Z",
  "uri": "https://www.example.com/products",
  "apiEventKey": "purchase_product",
  "cart": {
    "action": "purchase",
    "id": "CART-12345",
    "totalCartValue": 99.99,
    "items": [
      {
        "id": "PROD-001",
        "name": "Widget Pro",
        "quantity": 1,
        "unitPrice": "19.99"
      }
    ]
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Cart-Action Event

Cart-action events record actions that are related to adding, removing, and updating items in a cart. Each cart-action event corresponds to one item (product) in the cart.

Before you send this type of event, you must define these items in the user interface:

- The key for the event.
 - For the mobile channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Mobile App Action**. When you send the event, the event's payload must include the mobile event key from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event.
 - For the web channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Event API**. When you send the event, the event's payload must include the API event key from this definition.
- (Optional) Custom attributes. When you want to send custom attributes with an event, these attributes must exist in the respective cart-related event in the user interface.

These definitions enable the attributes to be used in SAS Customer Intelligence 360 and to be stored in the `properties_map_doc` field of the [CART_ACTIVITY_DETAILS](#) table.

Note: Custom attributes are available only at the cart level and not on the cart items.

This table describes the attributes for this event type:

Table 39 *Attributes for Cart-action Events*

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be <code>cartAction</code> .
cart	action	Required	String	CART_ACTIVITY_DETAILS.ACTIVITY_CD	Action taken by the user. Currently, the only action that is supported is <code>add</code> .
cart	id	Required	String	CART_ACTIVITY_DETAILS.CART_ID	ID of the cart. You must set either the <code>id</code> attribute or the <code>name</code> attribute.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
cart	name	Required	String	For informational purposes only	Name of the cart. You must set either the id attribute or the name attribute.
product	id	Required	String	CART_ACTIVITY_DETAILS.PRODUCT_ID	ID of the product. You must specify at least one of the following parameters: ■ productId ■ productName ■ productSku
product	name	Required	String	CART_ACTIVITY_DETAILS.PRODUCT_NM	Name of the product. You must specify at least one of the following parameters: ■ productId ■ productName ■ productSku
product	sku	Required	String	CART_ACTIVITY_DETAILS.PRODUCT_SKU	SKU of the product. You must specify at least one of the following parameters: ■ productId ■ productName ■ productSku
product	group	Optional	String	CART_ACTIVITY_DETAILS.PRODUCT_GROUP_NM	Group that the product belongs to. The maximum length for this value is 130 characters.
product	unitPrice	Optional	Number	CART_ACTIVITY_DETAILS.UNIT_PRICE_AMT	Price for a single unit of the cart item. This value must be an integer or a decimal.
product	availabilityMessage	Optional	String	CART_ACTIVITY_DETAILS.AVA	Message that is related to item availability. The maximum

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				ILABILITY_MESSAGE_TXT	length for this value is 650 characters.
product	savingsMessage	Optional	String	CART_ACTIVITY_DETAILED.SAVING_MESSAGE_TXT	Message that is related to discounts. The maximum length for this value is 650 characters.
product	shippingMessage	Optional	String	CART_ACTIVITY_DETAILED.SHIPPING_MESSAGE_TXT	Message that is related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.
product	quantity	Optional	Number	CART_ACTIVITY_DETAILED.QUANTITY_VALUE	Number of this item that has been added, removed, or updated in the cart. The value must be an integer or decimal.
properties	<custom property name>	Optional			Custom properties that are associated with the event. Make sure that the custom properties are defined for the event in the user interface.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a cart-action event:

```
{
  "eventName": "cartAction",
  "apiEventKey": "addBook",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/books/Mastering+Marketing+Data+Science%3A+A+Comprehensive+Guide+for+Today's+Marketers-p-9781394258727",
  "cart": {
    "action": "add",
    "id": "1234567890"
  },
}
```

```

    "product": {
      "name": "Mastering Marketing Data Science: A Comprehensive Guide for Today's
Marketers",
      "sku": "978-1-394-25872-7",
      "unitPrice": 60
    },
    "properties": {
      "Author": "Iain Brown",
      "type": "E-Book",
      "pages": "432",
      "releaseDate": "April 2024"
    }
  }
}

```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Click Event (Web Only)

Click events track when someone clicks an element on the page. Click events can also be used to trigger user-defined events in SAS Customer Intelligence 360, such as steps in a [business process event](#). Click events do not automatically populate a specific table in the Unified Data Model (UDM).

TIP Unless you need to send a raw event of this type, consider sending a [custom event on page 96](#) instead. Custom events enable more flexibility, because you send additional event data as custom attributes.

This table describes the attributes for this event type:

Table 40 Attributes for Click Events

Object	Attribute	Required or Optional	Data Type	Description
root	eventName	Required	String	Value must be click .
click	targetUri	Required	String	Destination URI that is associated with the event (that is, the HREF value of the click target). When you define an event in the user interface, this value corresponds to Target: href. Note: To store this value correctly, some characters must be HTML encoded when they are used as part of a URI. These are the only characters that can be unencoded: ■ alphanumeric characters

Object	Attribute	Required or Optional	Data Type	Description
				■ special characters: \$-_.+!*'() , ■ reserved characters: ;/?:@=& For more information, see the specification for URLs at https://www.ietf.org/rfc/rfc1738.txt.
click	altText	Optional	String	Value of the altText attribute of the click target.
click	anchorId	Optional	String	ID of an anchor. When you define an event in the user interface, this value corresponds to Target: id.
click	anchorName	Optional	String	Name of an anchor. When you define an event in the user interface, this value corresponds to Target: name.
click	elementTagName	Optional	String	Name of the element that was clicked. When you define an event in the user interface, this value corresponds to Target: tagName.
click	onClick	Optional	String	Value of the onClick attribute of the clicked element. When you define an event in the user interface, this value corresponds to Target: onClick.
click	targetInnerText	Optional	String	Inner text of the clicked element. When you define an event in the user interface, this value corresponds to Target: innerText.
click	targetSelectorPath	Optional	String	CSS selector path for the click target. When you define an event in the user interface, this value corresponds to Target: selectorPath.
<common attributes>		Optional		Common attributes on page 79 for all event types.

The following example shows a click event:

```
{
  "eventName": "click",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/en_us/solutions/customer-intelligence/marketing.html",
  "click": {
    "targetUri": "https://www.sas.com/en_us/solutions/customer-intelligence/
marketing/customer-journey-creation.html",
    "targetInnerText": "Explore journey creation"
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Click-Through Event

Click-through events, also known as spot-clicked events, indicate that a user has clicked on an element or link in the creative that is presented. Click-through events are often used as primary metric (conversion) or secondary metrics (milestones) in web tasks or mobile spot tasks.

Note: To use a click-through event as a primary metric or a secondary metric, you must first send a [impression-viewable event on page 101](#).

This table describes the attributes for this event type:

Table 41 Attributes for Click-Through Events

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be <code>spot_clicked</code> .
spot	id	Required	String	For internal use only	UUID for the spot that the content was shown in. For each spot, you must set either the id attribute or the key attribute.
spot	key	Required	String	For internal use only	Human-readable ID for the spot that content was shown in. For each spot, you must set either the id attribute or the key attribute.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
spot	taskId	Required	String	SPOT_CLICKED.task_id	Task ID that is associated with the spot that delivered content. This information is available from the response that you send to the Content Request API.
spot	creativeId	Required	String	SPOT_CLICKED.creative_id	ID of the creative that was delivered into the given spot. This information is available from the response that you send to the Content Request API.
spot	requestId	Optional	String	SPOT_CLICKED.request_id	Unique request ID, which can be used to relate records in the IMPRESSION_SPOT_VIEWABLE table and the IMPRESSION_DELIVERED table. This information is available from the response that you send to the Content Request API.
spot	targetUri	Optional	String	SPOT_CLICKED.url_txt	Link that was clicked in the creative.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a click-through event:

```
{
  "eventName": "spot_click",
  "channel": "mobile",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "/account/overview",
  "mobile": {
    "appId": "SASDemoApp"
  },
  "spot": {
    "key": "AccountOverviewBottom",
    "taskId": "602abd22-8da8-42df-9b90-8080a0703cdc",
    "creativeId": "f9e9d805-578c-45ea-8a6e-a25e9850a2ad",
    "requestId": "12808892-bfbe-4c42-9b20-8d28a4e09fce",
    "targetUri": "https://www.sas.com/en_us/solutions/customer-intelligence/marketing.html"
  }
}
```

```

    }
}

```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Custom Event

Custom events record any behavior that might not be associated with other event types. Custom events are more flexible than raw events (such as a click event) because they enable you to send any event data as custom attributes.

Custom events populate the [CUSTOM_EVENTS](#) table.

Before you send this type of event, you must define these items in the user interface:

- The key for the event.
 - For the mobile channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Mobile App Action**. When you send the event, the event's payload must include the mobile event key from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event.
 - For the web channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Event API**. When you send the event, the event's payload must include the API event key from this definition.
- (Optional) Custom attributes. When you want to send custom attributes with an event, these attributes must exist in the respective cart-related event in the user interface.

These definitions enable the attributes to be used in SAS Customer Intelligence 360 and to be stored in the `properties_map_doc` field of the [CUSTOM_EVENTS](#) table.

Table 42 *Attributes for Custom Events*

Object	Attribute	Required or Optional	Data Type	Description
root	eventName	Required	String	Value must be custom .
custom	name	Required	String	Name of the custom event. This value is how the event is displayed in the Custom Events report. You can use this field to store additional details. You might need to differentiate between custom events that share the same event key, but you can use different values for the custom name. For example, you might want to define a custom event for "Event Registration" and populate the custom name dynamically based on

Object	Attribute	Required or Optional	Data Type	Description
				the event type (like "registered for SAS Innovate 2026" or "registered for webinar XYZ").
custom	groupName	Required	String	Group name for the custom event, as you want it displayed in the custom event report. You can use this field to store additional details. You might need to differentiate between custom events that share the same event key, but you can use different values for the group name. For example, you might want to define a custom event for "Event Registration" and then populate the groupName attribute dynamically based on the event type (like "registered for SAS Innovate" or "registered for a webinar").
custom	revenue	Optional	Double	Revenue that might be associated with the custom event. For example: ■ This attribute enables you to determine the value of a campaign and not just a conversion rate. ■ For a donation-like event, this value could be the donation amount.
properties	<custom property name>	Optional		Custom properties that are associated with the event. Make sure that the custom properties are defined for the event in the user interface.
<common attributes>		Optional		Common attributes on page 79 for all event types.

The following example shows a custom event:

```
{
  "eventName": "custom",
  "channel": "mobile",
  "apiEventKey": "loanCalculator",
```

```

"clientTime": "2025-11-14T15:45:00Z",
"uri": "SASFinanceApp/home/tools",
"properties": {
  "Make": "Rolls Royce",
  "Model": "Cullinan",
  "Price": "450,000",
  "Term": "36"
},
"custom": {
  "groupName": "Tools",
  "name": "Car Loan Calculator"
},
"mobile": {
  "appId": "SASDemoApp"
}
}

```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Document Event (Web Only)

Document events record when users interact with document types on your site. These interactions are typically document downloads.

This table describes the attributes for this event type:

Table 43 *Attributes for Document Events*

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be document .
document	name	Optional	String	DOCUMENT_DETAILS.link_name	Name of the document that was downloaded. This value is stored in the link_name column in the DOCUMENT_DETAILS table.
document	id	Optional	String	DOCUMENT_DETAILS.link_id	ID of the document that was downloaded. The value is stored in the link_id column in DOCUMENT_DETAILS table.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
document	selectorPath	Optional	String	DOCUMENT_DETAILS.link_selector_path	Selector path of the document that was downloaded. This value is stored in the link_selector_path column in the DOCUMENT_DETAILS table. When the same document can be downloaded from more than one location on a single page, you can use this value to determine the actual location that was used.
document	altText	Optional	String	DOCUMENT_DETAILS.alt_text	Value of the altText attribute for the document that was downloaded. This value is stored in the alt_text column in DOCUMENT_DETAILS table.
document	targetUri	Required	String	DOCUMENT_DETAILS.uri_text	Link to the document that was downloaded.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a document event:

```
{
  "eventName": "document",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/en_us/gc/data-and-ai-impact-report.html",
  "document": {
    "name": "Data and AI Impact Report: The Trust Imperative",
    "targetUri": "https://www.sas.com/en_us/gc/data-and-ai-impact-report.html#download"
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Identity Events

Attach-Identity Event

Attach the visitor ID to the known identity in SAS Customer Intelligence 360 (such as `subject_id`, `customer_id` or `login_id`).

Make a request to an endpoint like this example:

```
POST https://<server-side host name>/t/events/d/<external tenant ID>/<visitor ID>/
id_type=<identity type>/id_value=<identity value>
```

The following table describes how to set the `id_type` and `id_value` query parameters:

URL Parameter	Description
<code>id_type</code>	The parameter must be set to one of these values: ■ <code>subject_id</code> ■ <code>login_id</code> ■ <code>customer_id</code>
<code>id_value</code>	the actual ID value.

The response repeats the values that you sent, like this example:

```
{
  "datahub_id": "5cbeb71a-5d33-3c24-8b25-01d539b46120",
  "visitor_id": "2d375c62893b8723940c2790",
  "id_type": "subject_id",
  "id_value": "olaf70001",
  "error": "No Errors"
}
```

If you receive a response with a 200 status value, the visitor ID is connected to the known ID that was sent in the request. For successful requests, you can ignore the body of the response.

After the identity is attached, all subsequent events that are sent with the same visitor ID are associated with the same known ID. This behavior continues until there is a [detach-identity request on page 100](#) or a different attach-identity request (which associates the visitor ID with a different known ID).

Detach-Identity Event

To disconnect a known identity ID from a visitor ID, send a DELETE request to the following URL:

```
DELETE https://<server-side host name>/t/events/d/<external tenant ID>/<visitor ID>
```

After this request, events that are sent with that visitor ID are treated as anonymous and no longer associated with a known ID. These events are anonymous until another [attach-identity request on page 76](#) occurs.

A status 200 confirms that the visitor ID has been disconnected from the known ID that was sent in the request. For successful requests, you can ignore the body of the response.

The response contains the new anonymous datahub_id, which is no attached to the visitor_id. This is an example of the response's body:

```
{
  "datahub_id": "4acd2f22-f85f-4c2d-9cb5-262adf59e53f",
  "visitor_id": "6bd1f9e40743da234d5da9d7",
  "error": "no"
}
```

Impression-Viewable Event

Impression-viewable (spot viewable) events track when a creative is displayed in your content. How you define “viewable” can be specific to your needs. For example, you might decide that a creative is viewable when a user scrolls past its position on the page.

Note: An impression-viewable event is required to identify conversions in the user interface. To generate conversions for tasks, make sure that you send an impression-viewable event before you send a [click-through event on page 94](#).. Conversions for a task are counted for users only when the users fulfill these requirements:

- 1 A user saw the content and generated an impresion-viewable event.
- 2 The user triggered the event that is defined as a primary or secondary metric in the corresponding task.

This table describes the attributes for this event type:

Table 44 *Attributes for Impression-Viewable Events*

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be spot_viewable .
spot	id	Required	String	For internal use only	UUID for the spot that the content was shown in. For each spot, you must set either the id attribute or the key attribute.
spot	key	Required	String	For internal use only	Human-readable ID for the spot that content was shown in. For each spot, you must set either the id attribute or the key attribute.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
spot	taskId	Required	String	IMPRESSION_SPOT_VIEWABLE.task_id	Task ID that is associated with the spot that delivered content. This information is available from the response that you send to the Content Request API.
spot	creativeId	Required	String	IMPRESSION_SPOT_VIEWABLE.creative_id	ID of the creative that was delivered into the given spot. This information is available from the response that you send to the Content Request API.
spot	requestId	Optional	String	IMPRESSION_SPOT_VIEWABLE.request_id	Unique request ID, which can be used to relate records in the IMPRESSION_SPOT_VIEWABLE table and the IMPRESSION_DELIVERED table. This information is available from the response that you send to the Content Request API.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a spot_viewable event:

```
{
  "eventName": "spot_viewable",
  "channel": "mobile",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "/account/overview",
  "mobile": {
    "appId": "SASDemoApp"
  }
}
```

```

    },
    "spot": {
      "key": "AccountOverviewBottom",
      "taskId": "602abd22-8da8-42df-9b90-8080a0703cdc",
      "creativeId": "f9e9d805-578c-45ea-8a6e-a25e9850a2ad",
      "requestId": "12808892-bfbc-4c42-9b20-8d28a4e09fce"
    }
  }
}

```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Internal-Search Event

Internal-search events record a user’s interactions with the search functionality of your site.

Before you send an internal-search event, you must define the event in the user interface:

- For the mobile channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Mobile App Action**. When you send the event, the event’s payload must include the mobile event key from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event.
- For the web channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Event API**. When you send the event, the event’s payload must include the API event key from this definition.

Table 45 Attributes for Internal-Search Events

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be <code>internalSearch</code> .
search	searchName	Required	String	SEARCH_RESULTS.search_nm	Name of the search used for grouping. For example, the value might be General Search , Product Search , or so on.
search	term	Required	String	SEARCH_RESULTS.search_phrase	Term that the user searched for. This value could be a phrase like Books about SAS Customer Intelligence .

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
search	fieldId	Required	String	SEARCH_RESULTS.search_field_id	ID of the field that the search term was entered in. You must set either the fieldId attribute or the fieldName attribute. When there are multiple fields in a search form, this value helps identify the field that the term was entered. For example, a search for a car type might have fields for make, model, year, mileage, and so on. You can use this attribute to associate the search term with the correct attribute.
search	fieldName	Required	String	SEARCH_RESULTS.search_field_name	Name of the field that the search term was entered in. You must set either the fieldId attribute or the fieldName attribute. When there are multiple fields in a search form, this value helps identify the field that the term was entered. For example, a search for a car type might have fields for make, model, year, mileage, and so on. You can use this attribute to associate the search term with the correct field for the car.
search	resultsDisplayed	Optional	Integer	SEARCH_RESULTS.search_results_displayed	Number of results returned by the given search.
search	resultsDisplayFlag	Optional	Boolean	SEARCH_RESULTS.results_displayed_flg	Boolean value (true or false) that specifies whether search results are displayed. When results are displayed, entries are recorded in the SEARCH_RESULTS_EXT table.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows an internal-search event:

```
{
  "eventName": "internalSearch",
  "apiEventKey": "globalSearch",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/en_us/solutions/customer-intelligence/marketing.html",
  "search": {
    "name": "Global Search",
    "term": "scheduled journey",
    "fieldName": "searchterm",
    "resultsDisplayed": 2740,
    "resultsDisplayFlag": true
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Load Event

This event indicates when a page loaded for a site, whether this corresponds to the content of an actual web page, a step in a single-page application (SPA), or a screen on a mobile device.

IMPORTANT You do not need to send load events explicitly, unless you want to record every load event that occurs. For each new URI in an event’s payload, SAS Customer Intelligence 360 generates a unique synthetic load event for that page. For more information, see [“Load Events and Sessions” on page 68](#).

This table describes the attributes for this event type:

Table 46 Attributes for Load Events

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be load
properties	referrer	Optional	String	PAGE_DETAILS.referrer_url_txt	URL that is associated with the current page
properties	pageTitle	Optional	String	PAGE_DETAILS.page_desc	Title of the page
<common attributes>		Optional			Common attributes on page 79 for all event types.

This is an example of a load event:

```
{
  "eventName": "load",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/en_us/solutions/customer-intelligence/marketing.html",
  "properties": {
    "pageTitle": "Customer Engagement Platform | SAS Customer Intelligence 360 | SAS",
    "ip": "149.173.160.3",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edge/142.0.0.0"
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Media Event (Web Only)

Media events track when a customer interacts with a media element. The interactions are “start”, “stop” and “end”. Be aware of these details about how to use these interactions:

- If a user skips a certain section of a media element you need to send a "stop" for the position the user was when they skipped and a "start" event for the position where the media element resumed.
- When the user pauses and resumes a media element you must send a "stop" and "start" event with the same position.
- The "end" action indicates that the end of the media element was reached without the user interacting with any controls.

This table describes the attributes for this event type:

Table 47 *Attributes for Media Events*

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be media .
media	action	Required	String	MEDIA_ACTIVITY_DETAILS.action	Type of action associated with the media. You can use one of these values: ■ start ■ stop ■ end
media	url	Required	String	MEDIA_ACTIVITY_DETAILS.media_uri_txt MEDIA_DETAILS.media_uri_txt MEDIA_DETAILS_EXT.media_uri_txt	URL that is associated with the media event. The maximum length for this value is 260 characters.
media	player	Required	String	MEDIA_DETAILS.media_player_nm	String or number that identifies the media player. The maximum length for this value is 30 characters.
media	id	Required	String	For internal use only	ID of the media element that is used to uniquely identify it.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
media	position	Required	Number	MEDIA_ACTIVITY_DETAILS.playhead_position	Playhead position in the media element where the action happened.
media	duration	Required	Number	MEDIA_DETAILS.media_duration_secs	Total duration of the media element, in seconds.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a media event:

```
{
  "eventName": "media",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/en_us/solutions/customer-intelligence/marketing.html",
  "media": {
    "action": "start",
    "url": "https://www.sas.com/44ee9cb6-0d13-4d50-b016-a7886c060835",
    "player": "html5",
    "id": "player-beb11276feab29ef6ba59256ce8d39d2_html5_api",
    "position": 0,
    "duration": 60
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Product-View Event

Product-view events track when a user is watching a certain product. This interaction could be on a product detail page or on a product overview page that contains multiple products.

TIP If multiple products are viewed on a single page, you can send all product-view events in a single payload.

Before you send this type of event, you must define these items in the user interface:

- The key for the event.
 - For the mobile channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Mobile App Action**. When you send the event, the event's payload must include the mobile event key from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event.
 - For the web channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Event API**. When you send the event, the event's payload must include the API event key from this definition.
- (Optional) Custom attributes. When you want to send custom attributes with an event, these attributes must exist in the respective cart-related event in the user interface.

These definitions enable the attributes to be used in SAS Customer Intelligence 360 and to be stored in the `properties_map_doc` field of the [PRODUCT_VIEWS](#) table.

Table 48 Attributes for Product-View Events

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be <code>productView</code> .
product	id	Required	String	PRODUCT_VIEWS.product_id	ID of the product. You must specify at least one of the following parameters: ■ id ■ name ■ sku
product	name	Required	String	PRODUCT_VIEWS.product_nm	Name of the product. You must specify at least one of the following parameters: ■ id ■ name ■ sku
product	sku	Required	String	PRODUCT_VIEWS.product_sku	SKU of the product. You must specify at least one of the following parameters:

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
					■ id ■ name ■ sku
product	group	Optional	String	PRODUCT_VIEWS.product_group_name	Group that the product belongs to. The maximum length for this value is 130 characters.
product	availabilityMessage	Optional	String	PRODUCT_VIEWS.availability_message_txt	Message that is related to item availability. The maximum length for this value is 650 characters.
product	unitPrice	Optional	Number	PRODUCT_VIEWS.price_val	Price for a single unit of the cart item. This value must be an integer or a decimal.
product	savingsMessage	Optional	String	PRODUCT_VIEWS.saving_message_txt	Message that is related to discounts. The maximum length for this value is 650 characters.
product	shippingMessage	Optional	String	PRODUCT_VIEWS.shipping_message_txt	Message that related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.
properties	<custom property name>	Optional			Custom properties that are associated with the event. Make sure that the custom properties are defined for the event in the user interface.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a product-view event:

```
{
  "eventName": "productView",
  "apiEventKey": "ViewBook",
  "channel": "web",
```

```

"clientTime": "2025-11-14T15:45:00Z",
"uri": "https://www.wiley.com/en-us/Mastering+Marketing+Data+Science%3A+A
+Comprehensive+Guide+for+Today's+Marketers-p-9781394258727",
"product": {
  "name": "Mastering Marketing Data Science: A Comprehensive Guide for Today's
Marketers",
  "sku": "978-1-394-25872-7",
  "unitPrice": 60
},
"properties": {
  "Author": "Iain Brown",
  "type": "E-Book",
  "pages": "432",
  "releaseDate": "April 2024"
}
}

```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Promotion Event

Promotion events record when users interact with your site's in-site advertising, offers, and campaigns. Examples of customer interaction can vary between a simple click-through to an action like using a promotion code during checkout.

Promotions are used to track an offer or in-site advertising that is not delivered by SAS Customer Intelligence 360. For offers that are delivered by SAS Customer Intelligence 360, you can track interactions through [impression-viewable on page 101](#) and [click-through on page 94](#) events.

Before you send a promotion event, you must define the event in the user interface:

- For the mobile channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Mobile App Action**. When you send the event, the event's payload must include the mobile event key from this definition and the Mobile Application ID from one of the mobile applications that are associated with the event.
- For the web channel, set the key in the user interface in **Events** ⇒ **<Event Type>** ⇒ **Event API**. When you send the event, the event's payload must include the API event key from this definition.

This table describes the attributes for this event type:

Table 49 Attributes for Promotion Events

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be promotion .
promotion	recordType	Required	String	Determines which tables the records are written to	Specifies how a user interacted with a promotion. Use one of these values: ■ display: indicates that a promotion was shown to a user. ■ use: indicates that a user interacted with a promotion. This could be a click or the usage of a discount code.
promotion	type	Required	String	PROMOTION_DISPLAYED.promotion_type_nm PROMOTION_USED.promotion_type_nm	Type of promotion. This value can be used to group your promotions, for example: ■ group promotions based on their source, like internal or external ■ group promotions based on the visual type, like banner, carousel, or popup
promotion	name	Required	String	PROMOTION_DISPLAYED.promotion_nm PROMOTION_USED.promotion_nm	Name of the promotion. This value is used to distinguish between promotions and describes the content of the promotion.
promotion	creative	Optional	String	PROMOTION_DISPLAYED.promotion_creative_nm PROMOTION_USED.promotion_creative_nm	Creative that is used in the promotion. This value could be the name of the image or a reference to a content ID. When you use different creatives for the same promotion, this value enables you to analyze the effectiveness of specific creatives.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
promotion	placementId	Optional	String	PROMOTION_DISPLAYED.promotion_placement_nm PROMOTION_USED.promotion_placement_nm	Placement of the promotion. This would be the actual location on the page.
promotion	trackingCode	Optional	String	PROMOTION_DISPLAYED.promotion_tracking_cd PROMOTION_USED.promotion_tracking_cd	Unique tracking code. This value enables you to link the promotion to additional data in your system (like internal revenue reports) or reports from external partners.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a promotion event:

```
{
  "eventName": "promotion",
  "channel": "web",
  "apiEventKey": "SASInnovate2026",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://www.sas.com/en_us/home.html",
  "promotion": {
    "recordType": "use",
    "type": "banner",
    "name": "SAS Innovate 2026",
    "placementId": "SAS Homepage",
    "trackingCode": "PLN62481_746023809"
  }
}
```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Submit Event (Web Only)

Submit events track when a customer submits a form. Submit events must include names, values, and number of submission attempts from at least one field in the form.

Submit events populate the [FORM_DETAILS](#) table. Every field that is listed in a `fields[]` object is written as a separate record in the FORM_DETAILS table.

Submit events can also be used to trigger user-defined events in SAS Customer Intelligence 360, such as steps in a [business process](#).

This table describes the attributes for this event type:

Table 50 *Attributes for Submit Events*

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
root	eventName	Required	String	For internal use only	Value must be submit .
form	id	Required	String	FORM_DETAILS.form_nm	ID of the form on the page or a human-readable value in the page's data that could be used to configure a step in a business process. You must set either the id attribute or the name attribute.
form	name	Required	String	FORM_DETAILS.form_nm	Name of the form on the page or a human-readable value in the page's data that could be used to configure a step in a business process. You must set either the id attribute or the name attribute.
form	attempts	Optional	String	FORM_DETAILS.attempt_index_cnt	The number of attempts (as a string) that have been made to submit the field value.
form	fields	Required	Array	The table fields that are	Array of objects, where each object defines a form field by name/value pairs.

Object	Attribute	Required or Optional	Data Type	Table Location (as Table.Column)	Description
				populated depend on the item that is sent: ■ name is sent to FORM_DETAIL.S.form_field_nm. ■ value is sent to FORM_DETAIL.S.form_field_value. ■ changes is sent to FORM_DETAIL.S.change_index_no.	You must include at least one form field in the array. Every element of the array contains these attributes: ■ name: name of the form field on the page or a human-readable field name that shows up in the data. You must set either the id attribute or the name attribute. ■ id: ID of the form field on the page or a human-readable field name that shows up in the data. You must set either the id attribute or the name attribute. ■ value: value the user has entered in the field. IMPORTANT You are responsible for obfuscating or encrypting sensitive values to prevent values from being stored in cleartext. ■ changes: number of attempts it took for the user to enter the final value.
<common attributes>		Optional			Common attributes on page 79 for all event types.

The following example shows a submit event:

```
{
  "eventName": "submit",
  "channel": "web",
  "clientTime": "2025-11-14T15:45:00Z",
  "uri": "https://innovate.sas.com/event/b8642a1b-9ad0-4af8-bc0f-cfac6a57724b/regProcessStep1?rp=e83b5216-6507-4675-8cb1-379238776683",
  "form": {
    "name": "SAS Innovate 2026 registration",
    "attempts": "2",
    "fields": [
      {
        "name": "firstName",
```

```

        "value": "Jane",
        "changes": 1
    },
    {
        "name": "lastName",
        "value": "Doe",
        "changes": 0
    },
    {
        "name": "organization",
        "value": "SAS Institute",
        "changes": 2
    }
]
}

```

Note: The example above is shown on multiple lines for readability, but you must send an event as a single line. For more information, see [“How to Send Events” on page 73](#).

Collecting Mobile Event Data

About Mobile Events

For information about working with mobile events, see [“Mobile Events” in SAS Customer Intelligence 360: User’s Guide](#) and event-related topics in [SAS Customer Intelligence 360: Developer’s Guide for Mobile Applications](#).

In [SAS Customer Intelligence 360: Developer’s Guide for Mobile Applications](#), code examples are provided for each event that is described in this section.

Add to Cart Events

An add to cart event is a cart-action event that tracks actions related to adding items to a cart. When an item is added to a cart, an add to cart event can be triggered.

Cart-action events populate the [CART_ACTIVITY_DETAILS](#) table.

Here are the parameters for this event type:

Table 51 Parameters for Add to Cart Events

Programmatic Name	Required or Optional	Data Type	Table Location	Description
productName	Optional	Character	CART_ACTIVITY_DETAILS.PRODUCT_NM	The name of the product. The maximum length for this value is 130 characters. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productId	Optional	Character	CART_ACTIVITY_DETAILS.PRODUCT_ID	The ID for the product. The maximum length for this value is 130 characters. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productSku	Optional	Character	CART_ACTIVITY_DETAILS.PRODUCT_SKU	The SKU value for the product. The maximum length for this value is 100 characters. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productGroup	Optional	Character	CART_ACTIVITY_DETAILS.PRODUCT_GROUP_NM	A group that the cart item belongs to. The maximum length for this value is 130 characters. For example, the value could be sporting goods.
productUnitPrice	Optional	Number	CART_ACTIVITY_DETAILS.UNIT_PRICE_AMT	The price for a single unit of the cart item. This value must be an integer or a decimal.

Programmatic Name	Required or Optional	Data Type	Table Location	Description
productQuantity	Optional	Number	CART_ACTIVITY_DETAILS.QUANTITY_VAL	The number of products that were added to the cart. The value must be an integer or decimal.
availabilityMessage	Optional	Character	CART_ACTIVITY_DETAILS.AVAILABILITY_MESSAGE_TXT	A message that is related to item availability. The maximum length for this value is 650 characters. For example, the value could be back-ordered.
shippingMessage	Optional	Character	CART_ACTIVITY_DETAILS.SHIPPING_MESSAGE_TXT	A message related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.
savingsMessage	Optional	Character	CART_ACTIVITY_DETAILS.SAVING_MESSAGE_TXT	A message related to discounts. The maximum length for this value is 650 characters.

Cart Events

Cart events contain general information about a cart such as the total value, tax, shipping cost, payment type, delivery type, and other shipping and billing information. Cart events can also contain cart item objects, which include information about individual items in a user's cart.

Purchase View, Checkout View and Cart View are examples of cart events.

Cart events populate the [ORDER_SUMMARY](#) table. Cart items that are embedded in the event populate the [ORDER_DETAILS](#) table.

Table 52 Parameters for Cart Events

Programmatic Name	Required or Optional	Data Type	Table Location	Description
cartId	Optional	Character	ORDER_SUMMARY.CART_ID	The customer's unique cart ID. The value must be less than or equal to 42 characters.

Programmatic Name	Required or Optional	Data Type	Table Location	Description
				When a customer creates a cart, completes a purchase, or cancels a cart, the same cart ID is associated with all of these actions. The cart_id can be used externally to associate this cart with your external commerce system. Ensure that ID values are unique for each customer's cart. Mobile SDK calls bypass the internal verification to ensure unique IDs for carts, so give each cart a unique ID to avoid duplicate data records.
orderId	Optional	Character	ORDER_SUMMARY. ORDER_ID	The customer's unique order ID. When a customer completes an order, the order ID is the unique identifier. Ensure that order IDs are unique for each customer's order. Mobile SDK calls bypass the internal verification to ensure unique IDs for orders, so give each order a unique ID to avoid duplicate data records.
totalCartValue	Optional	Number	ORDER_SUMMARY. TOTAL_PRICE_AMT	The total value for goods in the cart. The value must be an integer or a decimal.
tax	Optional	Number	ORDER_SUMMARY. TOTAL_TAX_AMT	The tax amount. The value must be an integer or a decimal.
shippingCost	Optional	Number	ORDER_SUMMARY. SHIPPING_AMT	The cost of shipping. The value must be an integer or decimal.
paymentType	Optional	Character	ORDER_SUMMARY. PAYMENT_TYPE_D ESC	The payment type. The maximum length for this value is 42 characters. For example, the value might be VISA or CHECK.

Programmatic Name	Required or Optional	Data Type	Table Location	Description
deliveryType	Optional	Character	ORDER_SUMMARY. DELIVERY_TYPE_D ESC	The delivery type. The maximum length for this value is 42 characters. For example, the value might be USPS or FEDEX.
shippingCity	Optional	Character	ORDER_SUMMARY. SHIPPING_CITY_NM	The city where the cart is being shipped. The maximum length for this value is 390 characters.
shippingRegion	Optional	Character	ORDER_SUMMARY. SHIPPING_STATE_R EGION_CD	The region where the cart is being shipped. The maximum length for this value is 256 characters. For example, in the United States this value could be the state.
shippingPostcode	Optional	Character	ORDER_SUMMARY. SHIPPING_POSTAL_ CD	The shipping code where the cart is being shipped. The maximum length for this value is 10 characters. For example, in the United States this value could be the ZIP code.
shippingCountry	Optional	Character	ORDER_SUMMARY. SHIPPING_COUNTR Y_NM	The country where the cart is being shipped. The maximum length for this value is 85 characters.
billingCity	Optional	Character	ORDER_SUMMARY. BILLING_CITY_NM	The city where the invoice should be sent. The maximum length for this value is 390 characters.
billingRegion	Optional	Character	ORDER_SUMMARY. BILLING_STATE_RE GION_CD	The region where the invoice should be sent. The maximum length for this value is 256 characters. For example, in the United States this value could be the state.

Programmatic Name	Required or Optional	Data Type	Table Location	Description
billingPostcode	Optional	Character	ORDER_SUMMARY. BILLING_POSTAL_CD	The billing code where the invoice should be sent. The maximum length for this value is 10 characters. For example, in the United States this value could be the ZIP code.
billingCountry	Optional	Character	ORDER_SUMMARY. BILLING_COUNTRY_NM	The country where the invoice should be sent. The maximum length for this value is 85 characters.
totalCartItems	Optional	Number	ORDER_SUMMARY. TOTAL_UNIT_QTY	An object that defines one or more cart items. The number is the total amount for the cart. The value must be an integer or a decimal. Note: Cart events can be sent with no cart items.

Cart items contain product information for each row in a cart. Product information includes these item details:

Table 53 *Parameters for Cart Items*

Programmatic Name	Required or Optional	Data Type	Table Location	Description
productName	Optional	Character	ORDER_DETAILS.PRODUCT_NM	The name of the cart item. The maximum length for this value is 130 characters. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productId	Optional	Character	ORDER_DETAILS.PRODUCT_ID	The ID of the cart item. The maximum length for this value is 130 characters. Note: You must define at least one of these parameters:

Programmatic Name	Required or Optional	Data Type	Table Location	Description
				■ productId ■ productName ■ productSku
productSku	Optional	Character	ORDER_DETAILS.PRODUCT_SKU	The SKU of the product. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productGroup	Optional	Character	ORDER_DETAILS.PRODUCT_GROUP_NAME	A group that the cart item belongs to. The maximum length for this value is 130 characters. For example, the value could be sporting goods.
productQuantity	Optional	Number	ORDER_DETAILS.QUANTITY_AMT	The number of this item that exists in the cart. The value must be an integer or decimal.
productUnitPrice	Optional	Number	ORDER_DETAILS.UNIT_PRICE_AMT	The price for a single unit of the cart item. This value must be an integer or a decimal.
availabilityMessage	Optional	Character	ORDER_DETAILS.AVAILABILITY_MESSAGE_TEXT	A message related to item availability. The maximum length for this value is 650 characters. For example, the value could be back-ordered.
shippingMessage	Optional	Character	ORDER_DETAILS.SHIPPING_MESSAGE_TEXT	A message related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.
savingsMessage	Optional	Character	ORDER_DETAILS.SAVINGS_MESSAGE_TEXT	A message related to discounts. The maximum length for this value is 650 characters.

Internal Search Events

Internal search events track users' interaction with the search functionality of your application.

Internal search events populate the [SEARCH_RESULTS](#) table.

Here are the parameters for this event type:

Table 54 *Parameters for Internal Search Events*

Programmatic Name	Required or Optional	Data Type	Table Location	Description
searchName	Optional	Character	SEARCH_RESULTS. SEARCH_NM	The name that is associated with the search.
searchTerm	Optional	Character	SEARCH_RESULTS. SRCH_PHRASE	The term that was used in the search.
resultsDisplayed	Optional	Number	SEARCH_RESULTS. SEARCH_RESULTS_ DISPLAYED	The number of search results that is returned.
resultsDisplayFlag	Optional	Boolean	SEARCH_RESULTS. RESULTS_DISPLAYE D_FLG	Boolean value (true or false) that specifies whether search results are displayed. When results are displayed, entries are recorded in the SEARCH_RESULTS_EXT table.
searchFieldId	Optional	Character	SEARCH_RESULTS. SRCH_FIELD_ID	The ID of the search field on your app.
searchFieldName	Optional	Character	SEARCH_RESULTS. SRCH_FIELD_NAME	The name of the search field on your app.

Product View Events

Product view events track when a customer views a single product on your application.

Product view events populate the [PRODUCT_VIEWS](#) table.

Here are the parameters for this event type:

Table 55 *Parameters for Product View Events*

Programmatic Name	Required or Optional	Data Type	Table Location	Description
productName	Optional	Character	PRODUCT_VIEWS.PRODUCT_NAME	The name of the product. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productId	Optional	Character	PRODUCT_VIEWS.PRODUCT_ID	The ID for the product. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productSku	Optional	Character	PRODUCT_VIEWS.PRODUCT_SKU	The SKU of the product. Note: You must define at least one of these parameters: ■ productId ■ productName ■ productSku
productGroup	Optional	Character	PRODUCT_VIEWS.PRODUCT_GROUP_NAME	A group that the cart item belongs to. The maximum length for this value is 130 characters. For example, the value could be sporting goods.
productUnitPrice	Optional	Number	PRODUCT_VIEWS.PRICE_VAL	The price for a single unit of the cart item. This value must be an integer or a decimal.
availabilityMessage	Optional	Character	PRODUCT_VIEWS.AVAILABILITY_MESSAGE_TXT	A message that is related to item availability. The maximum length for this value is 650 characters.

Programmatic Name	Required or Optional	Data Type	Table Location	Description
				For example, the value could be back-ordered.
shippingMessage	Optional	Character	PRODUCT_VIEWS.SHIPPING_MESSAGE_TXT	A message related to the shipping type, process, or special instructions. The maximum length for this value is 650 characters.
savingsMessage	Optional	Character	PRODUCT_VIEWS.SAVING_MESSAGE_TXT	A message related to discounts. The maximum length for this value is 650 characters.

Promotion Events

Promotion events track when users interact with your application's in-site advertising, offers, and campaigns. Customer interaction can be anything from a simple click-through to something else such as using a promotion code during checkout.

Promotion events populate the [PROMOTION_DISPLAYED](#) table and the [PROMOTION_USED](#) table. The record_type attribute determines whether it is a display event or a use event (which is a click-through), and records are written to the appropriate table.

Here are the parameters for this event type:

Table 56 Parameters for Promotion Events

Programmatic Name	Required or Optional	Data Type	Table Location	Description
recordType	Required	Character	Not applicable	How a user interacted with a promotion. The value must be set to use or display: ■ display: indicates that a promotion was shown to a user. ■ use: indicates that a user interacted with a promotion. This value determines which tables the records are written to.

Programmatic Name	Required or Optional	Data Type	Table Location	Description
promotionName	Optional	Character	PROMOTION_DISPL AYED.PROMOTION_ NM PROMOTION_USED. PROMOTION_NM	The promotion's name.
placementID	Optional	Character	PROMOTION_DISPL AYED.PROMOTION_ PLACEMENT_NM PROMOTION_USED. PROMOTION_PLAC EMENT_NM	An ID where the promotion's banner or link is located.
creative	Optional	Character	PROMOTION_DISPL AYED.PROMOTION_ CREATIVE_NM PROMOTION_USED. PROMOTION_CREA TIVE_NM	The name of the creative that is associated with the promotion.
trackingCode	Optional	Character	PROMOTION_DISPL AYED.PROMOTION_ TRACKING_CD PROMOTION_USED. PROMOTION_TRAC KING_CD	A campaign tracking code that is used to link to additional campaign information (such as cost, media agency used, and so on) in an external table.
promotionType	Optional	Character	PROMOTION_DISPL AYED.PROMOTION_ TYPE_NM PROMOTION_USED. PROMOTION_TYPE _NM	The promotion's type. Some examples of types are banner ads, organic search, pay-per-click, referrals, partner sites, affiliates, email, and newspaper articles.

Content Request API

About the Content Request API

IMPORTANT The developer documentation is available at [Content Request API](#).

Overview of the Content Request API

The Content Request API enables you to deliver content to your digital properties (such as websites, mobile apps, or ATMs) from your back-end systems, without requiring client-side implementations of the SAS tag or mobile SDK. This REST API provides you with more control over content delivery and the overall user experience while still enabling you to take advantage of the analytical and targeting capabilities of SAS Customer Intelligence 360.

Here are some benefits of implementing server-side content delivery:

- More control over content delivery and placement, enabling channel-agnostic delivery.
- Flicker reduction when content is displayed.
- Performance improvements. You can reduce your use of the SAS tag or mobile SDK, or combine the Content Request API with the Server-Side Events API and remove the SAS tag and mobile SDK.
- You can send data-only creatives that do not deliver content but impact the app's behavior.
- Provide your customers with specific offers through an on-premises interface that is configured to work with the Content Request API.

Depending on your use case, there are different ways that you can combine the server-side and client-side capabilities of SAS Customer Intelligence 360. For example, you might want to use the SAS tag for data collection but not for content delivery. Or, you might want to use features of SAS Customer Intelligence 360, such as identity management, but not use the SAS tag or mobile SDK for data collection or content delivery.

In either scenario, you can customize which parts of data collection and content delivery are performed by SAS Customer Intelligence 360 or explicitly by you in calls to server-side REST APIs.

Combining Content Requests with Server-Side Events versus Client-Side Events

The following steps describe a high-level overview of the differences in using the Content Request API with server-side event collection (through the Server-Side Events API) or client-side event collection with the SAS tag or mobile SDK.

- 1 Decide whether you want to use server-side requests with server-side event collection or client-side event collection.

- For a server-side only implementation, you have a website, a mobile app, or an external system that integrates with SAS Customer Intelligence 360. However, you do not want to use the SAS tag, JavaScript Events API, or mobile SDK.
 - For client-side event collection, you have a website, a mobile app that uses the SAS tag, or a mobile SDK for data collection. However, you want to deliver content directly from your servers. In this use case, you would first configure your site or mobile app as normal and implement the SAS tag or mobile SDK.
- 2 Define spots in the user interface for SAS Customer Intelligence 360, but do not specify the location of the spots.
 - a For web spots, define the **Spot key** field. For mobile spots, define the **Spot ID** field.
 - b (Optional) For each spot, define targeting attributes.

IMPORTANT For server-side implementations, you should send these attributes with the request so that SAS Customer Intelligence 360 has access to this information.

- 3 When a visitor requests a page in a browser or mobile app, your content management system (CMS) or eCommerce system makes a content request through the Content Request API. You can request data for up to 10 spots in a single call.
- 4 SAS Customer Intelligence 360 sends a JSON response that includes this information:
 - the content of the spot, such as HTML, text, a content reference, and so on
 - the spot or related items, including the task ID, creative ID, and spot ID
- 5 The CMS or eCommerce system delivers the content to the site or mobile app.
- 6 You must explicitly send spot-related events. When you send a request to the Content Request API, SAS Customer Intelligence 360 automatically records impression-delivered events and spot-requested events, but impression-viewable (spot-viewed) events and click-through (spot-clicked) events are not collected by default.

To record these events, you must send them explicitly based on your implementation:

- For server-side implementations, you must send the events through the [Server-Side Events API on page 67](#) with a visitor ID or a known identity type.

Note: For server-side content requests or server-side events, SAS does not verify that any values exist for a given tenant. SAS verifies only that the IDs are formatted like UUIDs.

- For implementations that also use the SAS tag or mobile SDK, send the events through the JavaScript Events API, the [SAS Mobile SDK](#), or the [Server-Side Events API on page 67](#).

For more information, see these topics:

- [“Send a Content Request for Identified \(Known\) Users” on page 132](#)
- [“Send a Content Request for Anonymous Users” on page 133](#)

Prerequisites for Using the Content Request API

If you do not already have a static JSON web token (JWT) or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an [event access point](#).

TIP When you create the access point, make note of the tenant ID. Use this value as the external tenant ID when you make calls to the REST API.

- 2 Create an authorization token. You can either [generate a static JWT](#) or use an [access token](#).
- 3 Make note of the value of the token. When you make calls to the REST API, this token is used in the Authorization header as a bearer token.

Send a Request to the Content Request API

How to Send a Content Request

Use the information in this section to send content requests based on the user's type. The following topics include more details about how to define properties based on different users (such as anonymous versus known IDs):

- [“Send a Content Request for Identified \(Known\) Users” on page 132](#)
- [“Send a Content Request for Anonymous Users” on page 133](#)

Note: This topic describes how to send a request using a POST call with a JSON body. However, you can send a request using a GET call and defining the spot details in the URL path. For more information about this method, see [Request content using URL path-based requests](#) in the *Content Request API for SAS Customer Intelligence 360* documentation.

To send a POST call for a content request, complete these steps:

- 1 Send a POST call to a URL like this example:

```
https://<server-side host name>/t/content/<external tenant ID>/id_type=<identity type>/<identity value>
```

Table 57 Content Request Path Variables

Variable Name	Description
<server-side host name>	Host name for your tenant, which is based on the region. Use one of the following host names: ■ United States: <code>i-us.ci360.marketing</code> ■ Europe: <code>i-eu.ci360.marketing</code> ■ Asia Pacific Northeast: <code>i-ap.ci360.marketing</code> ■ Sydney: <code>i-au.ci360.marketing</code> ■ Mumbai: <code>i-mu.ci360.marketing</code>
external tenant ID	Tenant ID that is used when the SAS tag is loaded on your website. The value can be found in the event access point that you created.
identity type	Type of identity that is used in the request. Select the value based on your use case: ■ Identified (known) users: Use <code>subject_id</code>, <code>login_id</code>, or <code>customer_id</code>. ■ Anonymous (unknown) users: □ When you are also using the SAS tag or mobile SDK, use <code>_ci360_id</code>. You can omit the identity value. □ For anonymous users in server-side only sessions, use <code>visitor_id</code> and set the identity value to the visitor ID.
identity value	Corresponding value for the identity type. This value is required when the identity type is <code>customer_id</code>, <code>login_id</code>, <code>subject_id</code>, or <code>visitor_id</code>. For new visitors (when the identity type is <code>_ci360_id</code>), you can leave the identity value empty.

- 2 In the body of the request, include JSON code to specify information for one or more spots. Use JSON code like this example:

```
{
  "spots": [
    {
      "spotid": "<spot ID for spot 1>",
      "<attribute1 name>": "<attribute1 value>",
      "<attribute2 name>": "<attribute2 value>"
    },
    {
      "spotkey": "<spot key for spot 2>",
      "<attribute1 name>": "<attribute1 value>"
    }
  ]
}
```

```
]
}
```

Note: You can send a maximum of 10 spots in a single request.

For each spot in the spots object, you can define the spot information based on the information in the following table:

Table 58 Spot Information

Property or Attribute Name	Description
spotkey spotid	Use either the spotkey property or the spotid property to identify the spot, but do not use both. ■ When you use spotkey, the value for <i>spot identifier</i> must be the human readable spot ID. This value is defined in SAS Customer Intelligence 360 on the spot's Spot Details page. □ For web spots, use the value in the Spot key field. □ For mobile spots, use the value in the Spot ID field. ■ When you use spotid, the value for <i>spot identifier</i> must be the UUID of the spot in SAS Customer Intelligence 360. In the user interface, the spot's UUID is part of the URL when you are viewing the spot. For example: <pre>https://<server>/SASWebMarketing/#!/spots/3bc28afe-7579-423f-a82d-766cafe57</pre>
attribute<n> name	Name of a spot attribute, as it is defined in SAS Customer Intelligence 360. In the user interface, the attributes are defined on the spot's Spot Attributes page. IMPORTANT Task targeting enables you to use session attributes (for example, device type, platform, and geo location) to determine eligibility for spot content. When you use the Content Request API, your tasks do not have access to built-in attributes such as device type and geo location. When you need these attributes to determine eligibility, you must define them as spot attributes and send them in your server-side content request.
attribute<n> value	Corresponding value for the attribute's name. This value should be specific to a given customer or

Property or Attribute Name	Description
	context so that it can be used for task targeting or rule-based content distribution.

See Also

[“Responses and Errors” on page 135](#)

Send a Content Request for Identified (Known) Users

Note: When the session uses the SAS tag (either the full tag or the JavaScript Events API) or the mobile SDK, see [“Send Content Requests for Augmented Sessions” on page 133](#).

When you make content requests for identified (known) users, send the request with an identity type and an identity value. By setting these values, you connect the impression events and spot events that are generated from server-side requests with client-side events that are generated by SAS Customer Intelligence 360.

Send a request based on the instructions in [“How to Send a Content Request” on page 129](#), but set the `<identity_type>` variable in the request to one of these values:

- `subject_id`
- `login_id`
- `customer_id`
- `visitor_id` (if the identity is attached to a known user)

IMPORTANT When you are sending content requests for sessions that also use the SAS tag or mobile SDK, make sure the identities match.

You can send an unknown ID value, which creates an identity in the system. However, this ID is not associated with the user’s visitor ID or device ID by default. To associate these IDs, you must send an attach-identity event through one of these methods:

- the SAS tag’s [JavaScript Events API on page 3](#)
- the [SAS Mobile SDK](#)
- an attach-identity event in the Server-Side Events API

See Also

[“Responses and Errors” on page 135](#)

Send a Content Request for Anonymous Users

Note: When the session uses the SAS tag (either the full tag or the JavaScript Events API) or the mobile SDK, see [“Send Content Requests for Augmented Sessions” on page 133](#).

When you make content requests for anonymous (unidentified) visitors for sessions that are only server side, send the request with a visitor ID.

- 1 Determine the visitor ID. You can use a visitor ID that is already available in your system, or you can request a visitor ID from SAS Customer Intelligence 360.

- **Using your own visitor ID:** make sure that your visitor ID complies with these rules:

- ☐ is a hexadecimal value that contains only numbers 0 to 9, the letters A through F, or the letters a through f
- ☐ is a minimum of eight characters long
- ☐ is a maximum of 128 characters long
- ☐ follows case-sensitive rules

- **Requesting a visitor ID:**

- 1 Send this call to request the visitor ID:

```
GET https://i-<region>.ci360.marketing/t/events/i/<external tenant ID>
```

The JSON response is similar to this example:

```
{ "visitorId": "<value>" }
```

- 2 Send a request based on the instructions in [“How to Send a Content Request” on page 129](#), but use these values in the URL:

- **id_type:** Set the value to `visitor_id`.
- **id_value:** Set the value to the visitor ID that you determined in the previous step.

IMPORTANT Use the same visitor ID for all events that are associated with a specific device or browser.

See Also

[“Responses and Errors” on page 135](#)

Send Content Requests for Augmented Sessions

When you send content requests for sessions that also use the SAS tag or mobile SDK, you can use the `_ci360_id` property to connect session information that is already being collected on the client side with server-side events (for example, impression events) that are generated from the Content Request API.

- 1 If the `_ci360_id` property already exists on the client side, retrieve the value through one of these methods:

- For the SAS tag, call this command:

```
ci360("info").ci360id
```

- For the mobile SDK, call the `getCi360Id()` method to retrieve the current value. For more information, see these links:

- [Android](#)
- [iOS](#)

Use the returned value in calls through the Content Request API.

- 2 For new users, or for when you cannot retrieve the `_ci360_id` property for a user, complete these steps:

- a Generate a new value for the `_ci360_id` property.

- 1 Set the `<identity type>` variable in the server-side request to `_ci360_id`.
- 2 Leave the value for `<identity value>` empty.
- 3 In the request's response, make note of the value that is returned for the `_ci360_id` property. Use this visitor ID for this user in subsequent requests to the Content Request API.

- b After you make a content request with an empty value for the `_ci360_id` property, synchronize the new value for the `_ci360_id` property. You must complete this step so that the client-side implementation (the SAS tag or mobile SDK) uses the anonymous identity that is generated server-side and does not generate a new one.

Use one of these methods based on the type of user:

- For the SAS tag, provide the value of the `_ci360_id` property through one of these sources:
 - a global JavaScript variable called `_ci360_id`
 - a URL query parameter named `_ci360_id`
 - a cookie named `_ci360_id`.

The SAS tag always checks these locations to see if a server-side value for `_ci360_id` should be used to overwrite a local value.

Alternatively, the SAS tag's JavaScript snippet can be modified to provide a specific location and name that the SAS tag should check for this value. For example:

```
ci360id:{source:'jsvar|cookie|url',
        name:'<corresponding name in the source that contains the _ci360_id>'}
```

- For the mobile SDK, set the value of the `_ci360_id` property by calling the `setCi360Id()` method. This method sets the device ID to the value that is assigned by the server. For more information, see these links:
 - [Android](#)
 - [iOS](#)

See Also

[“Responses and Errors” on page 135](#)

Responses and Errors

Response Body

When you send a request to the Content Request API, the response is JSON with the following payload:

Note: Depending on the request, the response contains only the `id_type` property or the `_ci360_id` property.

```
{
  "id_type": "string",
  "id_value": "string",
  "contents": [
    {
      "spot_id": "string",
      "spot_key": "string",
      "task_id": "string",
      "creative_id": "string",
      "variant_id": "string",
      "creative_id": "string",
      "request_id": "string",
      "error_msg": "string",
      "has_content": "boolean",
      "content": "string",
      "channel_type": "string"
    },
    {
      ...
    }
  ]
}
```

Use these values when you send impression-viewable events and click-through events through the [JavaScript Events API on page 3](#), [mobile SDK](#), or [Server-Side Events API on page 67](#).

The following table describes the items in the response:

Property Name	Description
id_type	Echoes the value that is passed in the request. You can use this value to verify the type from the request.
id_value	When the request includes a value set in the <code>id_value</code> property, the response echoes the value that is passed in the request. If the request is made with the <code>id_type</code> set to <code>_ci360_id</code> and the <code>id_value</code> is empty, the response sets the <code>id_value</code> property to the new <code>_ci360_id</code> value for this visitor.

Property Name	Description
	IMPORTANT For anonymous visitors, you must send this value to the client to synchronize event collection with the SAS tag or mobile SDK. In addition, for subsequent requests you must set the <code>_ci360_id</code> property to this value.
contents	Array with content and relevant IDs for each requested spot.
spot_id	UUID of the spot in SAS Customer Intelligence 360. In the user interface, the spot's UUID is part of the URL when you are viewing the spot. For example: <code>https://<server>/SASWebMarketing/#/spots/3bc28afe-7579-423f-a82d-766cafeefee57</code>
spot_key	Human readable spot ID that is defined in SAS Customer Intelligence 360 on the spot's Spot Details page. ■ For web spots, use the value in the Spot key field. ■ For mobile spots, use the value in the Spot ID field.
channel_type	Channel that is associated with the spot. This value can be web or mobile .
task_id	UUID of the task for which the user received the content in this spot. This ID must be provided when you send the impression-viewable event and click-through event. If the <code>task_id</code> is identical to the <code>spot_id</code> , the user was not eligible for any tasks defined with this spot. As a result, the delivered content is the default content that is defined in the spot.
creative_id	UUID of the creative that is returned for this spot. This ID must be provided when you send the impression-viewable event and click-through event.
variant_id	(Optional) When a spot is part of an A/B test, this value provides information about which variant of the A/B test was delivered. This ID needs to be provided when you send impression-viewable events and click-through events.
request_id	Unique ID that identifies the current instance of the impression. This ID is stored in the impression-delivered event, which is generated automatically by the Content Request API. TIP This ID should be provided when you send an impression-viewable event and click-through event, because the value enables you to relate impression-viewable events and click-through events to a specific impression that was delivered. When content is delivered through SAS Customer Intelligence 360, the session ID and detail ID (the instance of the page) link impressions to subsequent events. However, because the content was delivered outside of SAS Customer Intelligence 360, the impression-delivered event cannot send this information to any subsequent events.
error_msg	Error message, if there is an issue with the request.
has_content	Specifies whether the customer qualifies for content in the spot.

Property Name	Description
	■ true indicates that the content property contains a creative (HTML or text). The task_id column and creative_id column are populated. ■ false indicates that the user does not qualify for any content in this spot. The content property, the task_id property, and the creative_id property are empty.
content	Actual content of the creative. Depending on the creative type, this value could be HTML or plain text.

Response Codes

HTTP Status Code	HTTP Response	Description
200	JSON response. For more information, see “Responses and Errors” on page 135 .	The request was successful.
404	Tenant %s not found.	The tenant ID that was provided is not valid.
403	Feature is not enabled for the tenant %s on Server Side Content.	In early access, the feature for server-side content is available only for specific tenants.
400	Too many spots to be requested (#spot=%s, tenant=%s).	A single request can contain a maximum number of 10 spots.
404	No spots found from tenant (urlpath=%s, tenant=%s).	This response is received only when no spots in the request exist on the given tenant. If the request includes at least one valid spot, a 200 code is returned. For the remaining spots that are invalid, those sections in the JSON response have no content, but the sections contain an appropriate error message.
404	_ci360_id is invalid: %s.	The value that is provided for the _ci360_id property is invalid.

Error Messages

Error Message	Description
Spot uuid not found: %s	The spotid parameter provided in the request is not a valid spot ID.

Error Message	Description
Spot key not found: %s	The spotkey parameter provided in the request is not a valid spot key.
Not supported spot request: %s	The requested spot is not supported. You cannot submit a request that uses both the spot ID and the spot key.

Retrieve Spot Information

You can send a REST call to retrieve information about spots that are configured in SAS Customer Intelligence 360. Spots represent the locations where content is placed in your website or mobile app.

Note: If you define web spots in SAS Customer Intelligence 360 and do not set a spot key, the spot details are determined by the SAS tag. You can use these spots for [content requests on page 129](#), but you must use the `spot_id` attribute.

When you want to use only server-side requests (without including the SAS tag), you should define spots with a spot key and use those values in content requests.

List All Spot Definitions

To retrieve a list of all spot definitions, send a call to this URL:

```
https://<server-side host name>/t/spots/<external tenant ID>
```

The response contains the spot's UUID, spot key, and any defined attributes. For example:

```
{
  "spots": [
    {
      "spot_id": "36f547b5-2484-4e0b-8fd6-912005c4db79",
      "spot_key": "",
      "spot_name": "QMART - Spot 3 - bottom left",
      "spot_type": "web",
      "attributes": [
        {
          "name": "is_logged_in",
          "type": "character"
        }
      ]
    },
    {
      "spot_id": "d155fc09-59c8-4c0f-8f9d-bc733f188dbf",
      "spot_key": "DEMOWEB2",
      "spot_name": "Demo Web Spot 2",

```

```
    "spot_type": "web",
    "attributes": [
      {
        "name": "language",
        "type": "character"
      }
    ]
  },
  "page_size": 0,
  "page_num": 0,
  "specials": null
}
```

You can use the spot ID or spot key from the response when you [send a request for content on page 129](#).

Get a Paginated List of Spot Definitions

The base endpoint also supports two optional parameters that can be appended to the URL path:

Table 59 Path Parameters for Pagination

Parameter	Description
page_num	Enables pagination and sets which page number to return. Pagination uses a one-based index, so the first page is returned when page_num=1.
page_size	Specifies the number of results in the response.

To retrieve spot definitions, set the values for both the page_num parameter and page_size parameter. For example:

```
https://<server-side host name>/t/spots/<external tenant ID>/page_num=1/page_size=10
```

Use these guidelines when setting values for these parameters:

- To request help information, set the value for either page_size or page_num to 0.
- To retrieve spot definitions with a paginated response, set both page_num and page_size to values greater than 0. The values for page_size or page_num cannot be less than 0.

REST APIs

You can use the REST APIs to create and manage items in SAS Customer Intelligence 360. The REST APIs enable you to script processes that you perform regularly or access items on an as-needed basis.

Container Image Management API

About the Container Image Management API

IMPORTANT The developer documentation is available at [Container Image Management API](#).

Introduction to the Container Image Management API

The Container Image Management API enables you to list and pull images from the Amazon Elastic Container Registry (Amazon ECR). The API generates an authorization token and a bash command that you can use to authenticate your Docker client to the Amazon ECR. After you are authenticated, you can pull the images from the Amazon ECR.

Endpoints in the Container Image Management API

The root of the Container Image Management API is `/marketingContainerImage`. Combine the root with one of these endpoints:

- `/images` can be used to list the images from the Amazon ECR.
- `/token` can be used to generate an authorization token for a tenant. The returned token can be used to authenticate Docker clients to pull images from the Amazon ECR.

For examples of how to use these endpoints, see [“Examples for the Container Image Management API” on page 141](#).

Prerequisites to Use the Container Image Management API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see [“Setting Up Authorization” in SAS Customer Intelligence 360: Administration Guide](#).

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.

When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable *<external gateway host>* in the documentation.

- b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.
- 2 Generate the temporary JWT with your static JWT or the `client_jwt` value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the Container Image Management API

List Images

Introduction to the REST API for List Images

You can use this REST API to list images from the Amazon ECR. The API validates your tenant details and returns the images sorted from newest to oldest based on the most recent `imagePushedAt` timestamp.

Building the Base URL for the API Calls

To successfully use the REST API, start with the correct URL domain.

- 1 Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- 2 Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable *<external gateway host>* in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 3 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingContainerImage/images
```

Example: List Images

To list the images from the Amazon ECR:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 60 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to the /images endpoint:

GET https://<external gateway host>/marketingContainerImage/images?start=12&limit=14

This GET call can use these query parameters in the URL:

Table 61 *Parameters for List Images API*

Name	Required or Optional	Data Type	Description
start	Optional	Integer	This is the index of the first image to be returned. The default value is 0.
limit	Optional	Integer	The number of images to be returned beginning from the start index. The default value is 100.

Send the JWT token in the authorization header of the request. If the token is valid, the request continues.

This is an example of a successful response:

```
{
  "items": {
    "registryId": "string",
    "repositoryName": "string",
    "imageDigest": "string",
    "imageTags": [
      "string"
    ],
    "imageUri": [
      "string"
    ],
    "imageSizeInBytes": 0,
    "imagePushedAt": "string",
    "imageManifestMediaType": "string",
    "artifactMediaType": "string"
  },
}
```

```

    "count": 0,
    "start": 0,
    "limit": 0,
    "name": null
  }

```

This is an example of an error response:

```

{
  "message": "string",
  "id": "string",
  "errorCode": 0,
  "httpStatusCode": 0,
  "details": [
    "string"
  ],
  "remediation": "string",
  "errors": [
    {}
  ],
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "https://extapigwservice-<server>/<endpoint>/",
      "uri": "/<endpoint>",
      "type": "application/vnd.sas.collection"
    }
  ],
  "version": 0
}

```

When you have the list of images, you can use the token generation API to generate an authorization token and a bash command that you can use for authenticating Docker access to the Amazon ECR. After you are authenticated, you can use Docker commands to pull your images.

Token Generation

Introduction to the REST API for Token Generation

You can use this REST API to generate an authorization token and a bash terminal command that enables authentication for your tenant.

You can use the bash command that is returned by the `marketingContainerImage/token` endpoint to authenticate Docker access to the Amazon ECR. After you are authenticated, you can use Docker commands to pull your images.

Note: Before you can use Docker commands, make sure that Docker is installed on your system.

Building the Base URL for the API Calls

To successfully use the REST API, start with the correct URL domain.

- 1 Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.

- 2 Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external gateway host>/<endpoint>`

- 3 Use this base URL with the API paths to make calls. For example:

`GET https://<external gateway host>/marketingContainerImage/token`

Example: Generate Authorization Token

To generate an authorization token:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 62 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer <code>eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v</code>

- 2 Send a GET call to the `/token` endpoint:

`GET https://<external gateway host>/marketingContainerImage/token`

Send the JWT token in the authorization header of the request. If the token is valid, the request continues.

This is an example of a successful response:

```
{
  "version": 2,
  "tokenDetails": [
    {
      "authorizationToken": "QVdT0mV5SndZWGxz...",
      "expiresAt": "2025-03-07T06:21:38.611000+00:00",
      "proxyEndpoint": "https://952478859445.dkr.ecr.us-east-1.amazonaws.com",
      "dockerLoginBashCommand": "token='XXXXXXX' && echo $token | base64 --decode |
cut -d: -f2 | docker login --username $(echo $token | base64 --decode | cut -d: -f1)
--password-stdin https://952478859445.dkr.ecr.us-east-1.amazonaws.com"
    }
  ]
}
```

This is an example of an error response:

```
{
  "message": "string",
  "id": "string",
}
```

```

    "errorCode": 0,
    "httpStatusCode": 0,
    "details": [
      "string"
    ],
    "remediation": "string",
    "errors": [
      {}
    ],
    "links": [
      {
        "method": "GET",
        "rel": "self",
        "href": "https://<external_gateway_host>/<endpoint>/",
        "uri": "/<endpoint>",
        "type": "application/vnd.sas.collection"
      }
    ],
    "version": 0
  }
}

```

Authenticate Your Docker Client to the Amazon ECR by Using Authorization Token

When you get the ECR authorization token, you can pass it to the `docker login` command to authenticate your Docker client to the Amazon ECR. After you are authenticated, you can use Docker commands to pull the images.

You can authenticate with the ECR token by using one of these methods:

■ Linux:

```

token=<ECR authorization token>

echo $token | base64 --decode | cut -d: -f2 | docker login --username $(echo $token | base64
--decode | cut -d: -f1) --password-stdin <aws_account_id>.dkr.ecr.<region>.amazonaws.com

```

■ Windows PowerShell:

```

$token = "<ECR authorization token>"

$decodedToken =
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($token))
$username, $password = $decodedToken -split ":", 2

$password | docker login --username $username --password-stdin
"<aws_account_id>.dkr.ecr.<region>.amazonaws.com"

```

Pull Docker Images

To pull Docker images:

1 Authenticate Docker access to the Amazon ECR with either of the following options:

- Authenticate by using the `dockerLoginBashCommand` from the `/marketingContainerImage/token` response.
- Authenticate by using the authorization token from the `/marketingContainerImage/token` response. For more information, see [“Authenticate Your Docker Client to the Amazon ECR by Using Authorization Token”](#).

- 2 Run the following command to pull the Docker images:

```
docker pull <aws_account_id>.dkr.ecr.<region>.amazonaws.com/<tenant's repository name>:tag
```

Copy Item API

About the Copy Item API

IMPORTANT The developer documentation for this API is available at [Copy Item API](#).

Introduction to the Copy Item API

The Copy Item API provides a way to copy items from one environment (tenant) of SAS Customer Intelligence 360 to another environment. You can use the API to complete these tasks:

- package the items from an environment
- copy a promotion package from the source tenant to the target tenant
- check the status of promotion jobs

Endpoints in the Copy Item API

The root of the Copy Item API is `/marketingPromotion`. Combine the root with one of these endpoints:

- `/promotionPackageJob` creates a promotion package by copying items from a source tenant. You can specify whether the dependencies of items are included in the package.
- `/promotionJob` can be used with a POST call to submit a job that copies a promotion package into a target tenant.
- `/promotionJob/{promotionJobId}` retrieves the status of a promotion job.

For examples of how to use these endpoints, see [“Examples for the Copy Item API”](#) on page 147.

Prerequisites to Use the Copy Item API

If you do not already have a static JWT or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an access point so that you can generate a static JWT.

When you create the access point, make note of the **External gateway host** field on the Access Points page. This value is used as part of the URL for API calls that go through the external API gateway. Use this value when you see the variable *<external gateway host>* in the documentation.

- 2 (Optional) Create an API credential if you want to create a time-limited access token. This REST API does not require a temporary JWT, so you need to create an API credential only if you are completing the one-time setup to create an access token.
- 3 Make note of the value of the token that you need to use. The static JWT or access token must be specified in the Authorization header as a bearer token when you make calls to the REST API.

Examples for the Copy Item API

Create Promotion Package

Building the Base URL for the API Calls

To successfully use the Copy Item API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable *<external gateway host>* in the documentation.

An example might use the following URL for an endpoint:

`https://<external gateway host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

POST `https://<external gateway host>/marketingPromotion/promotionPackageJob`

Example: Create Promotion Package

Create a promotion package by copying items from a tenant. You can specify whether the dependencies of items are included in the package.

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 63 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a POST call to the /promotionPackageJob/ endpoint:

POST `https://<external_gateway_host>/marketingPromotion/promotionPackageJob`

In the body of the request, specify the following information:

```
{
  "objectId": "4ce1846f-468d-39d",
  "objectType": "creative",
  "dependencies": "true"
}
```

This is an example of a successful response:

```
{
  "promotionPackageId": "94722c8-43a3e-1a81c4",
  "objectId": "cdd1678-47ff-bafb",
  "type": "creative",
  "name": "2B",
  "folderPath": null,
  "code": "TSK_35",
  "status": "designing",
  "createdTimeStamp": 1636995305604,
  "dependencies": true
}
```

Copy Promotion Package

Building the Base URL for the API Calls

To successfully use the Copy Item API, start with the correct URL domain.

- 1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external_gateway_host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

POST `https://<external_gateway_host>/marketingPromotion/promotionJob`

Example: Copy Promotion Package

Create a job to copy the items in an existing promotion package to the target tenant (the tenant that submits the request).

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 64 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a POST call to the /promotionJob/ endpoint:

POST `https://<external_gateway_host>/marketingPromotion/promotionJob`

In the body of the request, specify the following information:

```
{
  "promotionPackageId": "8d5c86-41a-2b6392",
  "businessContextId": "475a7d-41d-a422e06"
}
```

This is an example of a successful response:

```
{
  "promoteJobId": "8b3f71-44e-012c5d",
  "packageId": "6ff03-55f-99f4ba",
  "createdTimeStamp": 1637075863619
}
```

Retrieve Status of Promotion Job

Building the Base URL for the API Calls

To successfully use the Copy Item API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.

- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingPromotion/promotionJob/<promotionJobID>
```

Example: Retrieve Status of Promotion Job

Return the status of a promotion job for a package.

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 65 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGci0.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a GET call that includes the ID for the promotion job to the `/promotionJob/` endpoint. For example, this call submits a query for a promotion job with an ID of `1651b4-42f-92a106`.

```
GET https://<external gateway host>/marketingPromotion/promotionJob/1651b4-42f-92a106
```

This is an example of a successful response:

```
{
  "promotionJobId": "1651b4-42f-92a106",
  "promotionPackageId": "db672a-4ca1-83db92",
  "objectType": "creative",
  "businessContextId": "757d7d-410-342e09",
  "status": "SUCCESS",
  "promotionErrors": null,
  "promotionWarnings": [
    "The code-format setting for creative is automatic, so an external code was generated for \"2B\"."
  ],
  "promotionInfo": null
}
```

Decisioning Subject Contact API

About the Decisioning Subject Contact API

IMPORTANT The developer documentation is available at [Decisioning Subject Contact API](#).

Introduction to the Decisioning Subject Contact API

The Decisioning Subject Contact API in SAS Customer Intelligence 360 enables you to upload contact and response data for actions that are delivered to customers after decisions are executed. The REST API is exposed by software provided by SAS that customers deploy and operate within their own environment as a service. The service integrates with a PostgreSQL database.

Use this API to perform the following tasks:

- record when an action is presented to a customer
- capture customer responses to those actions
- track which actions have been successfully contacted

Endpoints in the Decisioning Subject Contact API

The root of the Decisioning Subject Contact API is `/marketingDecisioningSubjectContact`. Combine the root with one of these endpoints:

- `/responses` uploads multiple responses in a single request.
- `/contacts` uploads multiple contacts in a single request.
- `/markAsContacts` marks multiple actions as contacted in a single request.

Prerequisites for Using the Decisioning Subject Contact API

If you use a Helm chart to provision a PostgreSQL database instance as part of the decision deployment, there are no prerequisites for using the Decisioning Subject Contact API.

As part of the deployment, the following actions occur:

- PostgreSQL is deployed within the Kubernetes cluster.
- The PostgreSQL deployment, including the database, schema, and persistence, is managed through Helm values.
- The required database schema and tables are automatically created during deployment.

Before you deploy the Helm chart, update the values.yaml file with the required settings. For more information, see [“Deploying Native Components by Using Public Helm Charts” in SAS Customer Intelligence 360: Administration Guide](#).

If you use an externally managed PostgreSQL database, complete these steps to access the Decisioning Subject Contact API:

- 1 Create a Relational Database Service (RDS) instance and set up a database. For more information, see <https://www.postgresql.org/>.
- 2 Clone the following GitHub repository: <https://github.com/sassoftware/ci360-helm-charts/>
- 3 Run the CI360_ContactHistory_DDL_Scripts.sql script to create the contactresponse schema and the required tables. The file is available at <https://github.com/sassoftware/ci360-helm-charts/tree/main/tools/marketing-decisioning>.

Examples for the Decisioning Subject Contact API

Add Response Data

Introduction to the REST API for Adding Response Data

Use the /responses endpoint to upload response data for actions that are delivered to customers. This endpoint enables you to upload data about customer engagement, such as whether a customer accepts or rejects an offer. For example, a customer receives an offer on a website and clicks Yes to accept it. You use this endpoint to send a response event.

Example: Add Response Data

To upload responses, send a POST call to this endpoint:

```
https://<on-premises server>/marketingDecisioningSubjectContact/responses
```

Note: This REST API is hosted on premises. Contact your administrator if you need more information about the domain.

In the body of the request, specify the following information:

```
{
  "items": [
    {
      "identityValue": "1111",
      "identityType": "customer_id",
      "subjectLevel": "customer",
      "responseDateTime": "2025-03-10T11:22:18Z",
      "responseChannel": "Web",
      "responseTrackingCode": "3b147bb8-c2d0-496f-bc26-b45554bd7334",
      "responseTxt": "No",
      "responseTypeCode": "0001",
      "responseTypeTxt": "WebResponse",
      "actionCode": "promo-email-001",
    }
  ]
}
```

```

        "actionCategory": "cross sell",
        "actionType": "promotion"
    },
    {
        "identityValue": "2222",
        "identityType": "customer_id",
        "subjectLevel": "customer",
        "responseDateTime": "2025-03-10T11:22:18Z",
        "responseChannel": "ATM",
        "responseTrackingCode": "e8c2a929-467c-ef43-815e-9ad5bf0f00b4",
        "responseTxt": "Yes",
        "responseTypeCode": "0002",
        "responseTypeTxt": "ATMResponse",
        "actionCode": "promo-email-001",
        "actionCategory": "cross sell",
        "actionType": "promotion"
    }
]
}

```

This is an example of a successful response:

```

{
  "count": 2,
  "skipped": 0
}

```

Add Contact Data

Introduction to the REST API for Adding Contact Data

Use the /contacts endpoint to upload contact data for actions that are delivered to customers. This endpoint enables you to upload the context of the interaction, such as the channel, action details, and customer identity.

Example: Add Contact Data

To upload contacts, send a POST call to this endpoint:

```
https://<on-premises server>/marketingDecisioningSubjectContact/contacts
```

Note: This REST API is hosted on premises. Contact your administrator if you need more information about the domain.

In the body of the request, specify the following information:

```

{
  "items": [
    {
      "identityValue": "1111",
      "identityType": "customer_id",
      "subjectLevel": "customer",
      "channelName": "web",
      "actionCode": "ext-code-123",
      "actionCategory": "cross sell",
      "actionType": "offer",

```

```

        "responseTrackingCode": "cc3bacbe-f016-4a2c-9682-e838d45efdf1",
        "presentedTxt": "Offered through Web channel",
        "presentationTimeStamp": "2026-03-25T10:30:00Z"
    },
    {
        "identityValue": "2222",
        "identityType": "customer_id",
        "subjectLevel": "customer",
        "channelName": "atm",
        "actionCode": "ext-code-123",
        "actionCategory": "cross sell",
        "actionType": "offer"
    }
]
}

```

This is an example of a successful response:

```

{
    "count": 2,
    "skipped": 0,
    "items": [
        {
            "identityValue": "1111",
            "identityType": "customer_id",
            "subjectLevel": "customer",
            "channelName": "web",
            "actionCode": "ext-code-123",
            "actionCategory": "cross sell",
            "actionType": "offer",
            "responseTrackingCode": "cc3bacbe-f016-4a2c-9682-e838d45efdf1",
            "presentedTxt": "Offered through Web channel",
            "presentationTimeStamp": "2026-03-25T10:30:00Z"
        },
        {
            "identityValue": "2222",
            "identityType": "customer_id",
            "subjectLevel": "customer",
            "channelName": "atm",
            "actionCode": "ext-code-123",
            "actionCategory": "cross sell",
            "actionType": "offer",
            "responseTrackingCode": "5f6d2b9a-7c41-4e8a-9d13-2a6e4c1b8f52",
            "presentationTimeStamp": "2026-05-28T10:30:00Z"
        }
    ]
}

```

Mark Actions as Contacted

Introduction to the REST API for Marking Actions as Contacted

Use the /markAsContacts endpoint to mark actions that are delivered to customers as contacted. Use this endpoint when the actual delivery happens outside the system and when you need to

update the contact status afterward. For example, an external system sends a mobile notification. After delivery, you call this endpoint to mark the action as contacted.

Example: Mark Actions as Contacted

To mark multiple actions as contacted in a single request, send a POST call to this endpoint:

```
https://<on-premises server>/marketingDecisioningSubjectContact/markAsContacts
```

Note: This REST API is hosted on premises. Contact your administrator if you need more information about the domain.

In the body of the request, specify the following information:

```
{
  "items": [
    {
      "responseTrackingCode": "e241c847-3e32-4b7e-ab9d-fc20c96134bf",
      "presentedTxt": "Offered through Web channel",
      "presentationTimeStamp": "2026-04-16T10:30:00Z"
    },
    {
      "responseTrackingCode": "afc0f062-8366-44ab-b20e-13a4dcfcc20b",
      "presentedTxt": "Offered through Mobile channel"
    }
  ]
}
```

This is an example of a successful response:

```
{
  "count": 2,
  "skipped": 0
}
```

Marketing Decisioning API

About the Marketing Decisioning API

IMPORTANT The developer documentation is available at [Marketing Decisioning API](#).

Introduction to the Marketing Decisioning API

The Marketing Decisioning API in SAS Customer Intelligence 360 provides the following capabilities:

- returns decisions that have been marked Ready at least once. This capability enables you to retrieve a deployment-ready list of decisions without additional filtering or manual selection.

- returns information about all published versions, including version numbers, image URLs, image digests, version names, and timestamps. This information helps you identify the latest published version and review the publication history of a decision.
- returns decision metadata such as the decision name, decision ID, decision state, and identity type.
- returns all image information that is required for deployment. This information includes the fully qualified URL image that is used to pull the image from the container registry and the SHA-256 image digest that is used for verifying image integrity after the image is pulled.

Endpoints in the Marketing Decisioning API

The root of the Marketing Decisioning API is `/marketingDecisioning`. The following endpoint is available:

- `/decisionImages` - returns a list of images for decisions that have been marked Ready at least once.

For more information about how to use this endpoint, see [“Examples for the Marketing Decisioning API”](#).

Prerequisites for Using the Marketing Decisioning API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see [“Setting Up Authorization” in SAS Customer Intelligence 360: Administration Guide](#).

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.

When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable `<external gateway host>` in the documentation.
 - b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.
- 2 Generate the temporary JWT with your static JWT or the `client_jwt` value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the Marketing Decisioning API

About List Decision Images

Introduction to the REST API for Listing Decision Images

Use the `/decisionImages` endpoint to retrieve a list of decision images for decisions that have been marked Ready at least once. Each item in the response represents a decision and includes the publication history of all decision images that were generated for that decision. For each image, the response provides information such as the image URL, image digest, version name, and the timestamp.

You can use this endpoint to determine which decision image should be deployed for a decision and to review the complete publication history of a decision.

Build the Base URL for API Calls

Before making API calls, determine the correct URL domain.

- 1 Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- 2 Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external_gateway_host>/<endpoint>`

- 3 Use this base URL with the API paths to make calls. For example:

GET `https://<external_gateway_host>/marketingDecisioning/decisionImages`

Example: List Decision Images

To retrieve a list of decision images for decisions that have been marked Ready:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 66 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer <code>eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v</code>

2 Send a GET call to the /decisionImages endpoint:

GET `https://<external_gateway_host>/marketingDecisioning/decisionImages`

Send the JWT token in the authorization header of the request. If the token is valid, the request continues.

This is an example of a successful response:

```
{
  "links": [],
  "name": "Decision Images",
  "accept": "application/vnd.sas.marketing.decisioning.decisionImages+json",
  "start": 0,
  "count": 1,
  "items": [
    {
      "decisionId": "f19b5168-b21e-48e5-9ee8-707a106f10f1",
      "decisionName": "NBA Decision",
      "decisionState": "ready",
      "identityType": "customer_id",
      "versions": [
        {
          "version": "V2",
          "imageUrl": "952478859445.dkr.ecr.us-east-1.amazonaws.com/tenant-images-acdecama:mktdecmorethanonepublish2026063004553164",
          "user": "user@example.com",
          "versionName": "mktdecmorethanonepublish2026063004553164",
          "timestamp": "2026-06-30T04:56:59.153587Z",
          "imageDigest": "sha256:5aab7c1a19770fe9912f382c2e0a8c02fad97195c9ffc7e4675e8aa16d88ddb0"
        },
        {
          "version": "V1",
          "imageUrl": "952478859445.dkr.ecr.us-east-1.amazonaws.com/tenant-images-acdecama:mktdecmorethanonepublish2026063004511348",
          "user": "user@example.com",
          "versionName": "mktdecmorethanonepublish2026063004511348",
          "timestamp": "2026-06-30T04:52:41.26269Z",
          "imageDigest": "sha256:112ee7785f709ea928e0efe60c108797c11317f75d718ca65fd1ce3970f1c56c"
        }
      ]
    }
  ],
  "limit": 5,
  "version": 2
}
```

This is an example of a failed response:

```
{
  "message": "string",
  "id": "string",
  "errorCode": 0,
  "httpStatusCode": 0,
  "details": [
    "string"
  ],
}
```

```

    "remediation": "string",
    "errors": [
        {}
    ],
    "links": [
        {
            "method": "GET",
            "rel": "self",
            "href": "https://extapigwservice-<server>/<endpoint>/",
            "uri": "/<endpoint>",
            "type": "application/vnd.sas.collection"
        }
    ],
    "version": 0
}

```

After retrieving the list of decision images, use the image URL and image digest that is returned in the response to pull the appropriate container image and verify that the image matches the expected decision version.

Digital Assets API

About the Digital Assets API

IMPORTANT The developer documentation for this REST API is available at [Digital Assets API](#).

Introduction to the Digital Assets API

The Digital Assets API enables you to access, use, manage, and store digital assets in a centralized, web-based repository in SAS Customer Intelligence 360.

This API enables you to manage your assets in different ways:

- perform basic functions such as create, view, update, copy, move, share, and delete
- preview an asset or create a thumbnail (without needing tools like Adobe Illustrator or Corel Draw)
- create versions and renditions
- download, upload, and store assets
- search for and retrieve assets based on user-defined tags
- share assets for reuse in a web browser

Endpoints in the Digital Assets API

The root of the Digital Assets API is `/marketingDigitalAssets`. Combine the root with one of the following endpoints:

- `/assetJobs` contains operations related to jobs for digital assets. For example, you can create jobs that create digital assets, catalogs digital assets, or update digital assets.
- `/attributeFile` contains operations to upload standard properties and custom properties for digital assets.
- `/digitalAssets` contains operations to create, manage, and update digital assets. This endpoint also includes descendant endpoints that enable you to access operations for renditions and revisions for a specified digital asset.
- `/folders` contains operations to create, manage, and update folders that organize digital assets.

For examples of how to use these endpoints, see [“Examples for the Digital Assets API” on page 160](#).

Prerequisites for Accessing the Digital Assets API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see [“Setting Up Authorization” in SAS Customer Intelligence 360: Administration Guide](#).

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.

When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable `<external gateway host>` in the documentation.
 - b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.
- 2 Generate the temporary JWT with your static JWT or the `client_jwt` value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the Digital Assets API

Digital Assets

About the REST API Endpoint for Digital Assets

To access, retrieve, and manage digital assets through the REST API, use the endpoint /digitalAssets. You can perform the following actions:

- create an asset
- retrieve an asset or a collection of assets
- update an asset
- get the current revision or a specified revision of an asset

The developer documentation for this REST API is available at [Digital Assets API](#).

Building the Base URL for the API Calls

To successfully use the Digital Assets API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external gateway host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

GET `https://<external gateway host>/marketingDigitalAssets/digitalAssets`

Example: Create a Digital Asset

This example lists the steps to create a new digital asset file:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 67 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer <code>eyJ0eXAiOiOGci0uDAxMGFhODEJ9.BRv_nD6v</code>

- 2 Send a POST call to create a new digital asset file:

POST https://<external_gateway_host>/marketingDigitalAssets/digitalAssets

This is an example of a request body:

```
{
  "name": "Email Banner",
  "description": "Description for email banner",
  "parentFolderId": "A5B1BT2C-AA00001L~Tree",
  "expiredTimeStamp": "2020-06-29T13:00:00Z",
  "domainObjectTypeCode": "Marketing Product Category"
}
```

3 This is an example of a successful response:

```
{
  "version": 1,
  "id": "80b44d55-41ca-4bac-8843-2e01e151b9dd",
  "name": "Email Banner",
  "description": "Description for email banner",
  "createdBy": "sasuser",
  "creationTimeStamp": "2019-07-22T12:25:37Z",
  "modifiedBy": "sasuser",
  "modifiedTimeStamp": "2019-07-27T01:27:51Z",
  "state": "active",
  "locked": false,
  "lockedTimeStamp": null,
  "lockedBy": null,
  "downloadDisabled": false,
  "downloadDisabledTimeStamp": null,
  "downloadDisabledBy": null,
  "expired": false,
  "expiredTimeStamp": "2020-06-29T13:00:00Z",
  "subType": "defaultAsset",
  "parentFolderId": "A5B1BT2C-AA00001L~Tree",
  "parentFolderPath": "All Assets/Banners",
  "downloadCount": 1,
  "ratingCount": 0,
  "ratingTotal": 0,
  "ratingAverage": 0,
  "missingMetadata": false,
  "active": true,
  "permission": "write",
  "publicUrlAvailable": "No",
  "localizedPublicUrlAvailable": "No",
  "publicUrl": null,
  "localizedExternalizationError": null,
  "relatedSource": null,
  "revisionNumber": 1,
  "revisionComment": "Revision in Banner",
  "revisionId": "593e40e4-3735-4e3a-978c-f68a6c7900be",
  "currentRevision": true,
  "domainObjectTypeCode": "Marketing Product Category",
  "customAttributes": {},
  "tags": [
    {
      "id": "ada02707-bc4f-447e-9f8f-d51f953dbc9a",
      "name": "DigitalBanner",
      "createdBy": "rdt0007",
```

```

    "createdDate": "2019-07-30T08:41:33.155Z",
    "lastModifiedBy": "sasuser",
    "lastModifiedDate": "2019-07-30T08:41:33.155Z",
    "count": 0,
    "links": [
      {
        "method": "GET",
        "rel": "self",
        "href": "https://<external gateway host>/marketingPlan/tags/ada02707-
bc4f-447e-9f8f-d51f953dbc9a",
        "uri": "/tags/ada02707-bc4f-447e-9f8f-d51f953dbc9a",
        "type": "application/vnd.sas.marketing.tag"
      }
    ]
  },
  ],
  "identityPermissions": [
    {
      "name": "sasuser",
      "displayName": "Demo User14",
      "type": "user",
      "permission": "write"
    },
    {
      "name": "sasuser",
      "displayName": "Demo User7",
      "type": "user",
      "permission": "write"
    },
    {
      "name": "sasuser",
      "displayName": "Demo User1",
      "type": "user",
      "permission": "no_access"
    },
    {
      "name": "sasuser",
      "displayName": "Demo User5",
      "type": "user",
      "permission": "read"
    }
  ],
  "attributeGroups": [],
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd",
      "type": "application/vnd.sas.marketing.asset"
    },
    {
      "method": "PUT",
      "rel": "update",

```

```

        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd",
        "type": "application/vnd.sas.marketing.asset"
    },
    {
        "method": "PATCH",
        "rel": "patch",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd",
        "type": "application/vnd.sas.marketing.asset"
    },
    {
        "method": "GET",
        "rel": "getDigitalAssets",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets",
        "uri": "/digitalAssets",
        "type": "application/vnd.sas.collection"
    },
    {
        "method": "GET",
        "rel": "parent",
        "href": "https://<external gateway host>/marketingDigitalAssets/folders/
A5V62Q40-AA0003GA~Tree",
        "uri": "/folders/A5V62Q40-AA0003GA~Tree",
        "type": "application/vnd.sas.marketing.folder"
    },
    {
        "method": "GET",
        "rel": "revisions",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions",
        "type": "application/vnd.sas.collection"
    },
    {
        "method": "POST",
        "rel": "createRevision",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions",
        "type": "application/vnd.sas.marketing.asset.revision"
    },
    {
        "method": "GET",
        "rel": "currentRenditions",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/1/renditions",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/1/
renditions",
        "type": "application/vnd.sas.collection"
    },
    {
        "method": "POST",

```

```

        "rel": "createRendition",
        "href": "https://<external_gateway_host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/-1/renditions",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/-1/
renditions",
        "type": "application/vnd.sas.marketing.asset.rendition"
    }
]
}

```

Digital Assets Folders

About the REST API Endpoint for Digital Assets Folders

To access and retrieve digital assets in a folder through the REST API, use the Digital Assets API endpoint `/folders`. You can perform the following actions:

- retrieve a folder or a list of folders that contain digital assets
- create, modify, or delete one or more folders that contain digital assets

The developer documentation for this REST API is available at [Digital Assets API](#).

Building the Base URL for the API Calls

To successfully use the Digital Assets API, start with the correct URL domain.

1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external_gateway_host>/marketingDigitalAssets/digitalAssets
```

Example: Retrieve Digital Assets in a Folder

This example demonstrates how to retrieve a list of digital assets in a folder. The example retrieves the list of folders first so you can find the folder ID that you need. Then, you use the folder ID to retrieve the list of assets that are members of that folder.

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 68 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the list of folders:

- a In the request header, set the Accept parameter to `application/vnd.sas.collection+json;version=1`.

- b Use this GET call:

GET `https://<external_gateway_host>/marketingDigitalAssets/folders`

In the response, make note of the folder's ID value. The response is similar to this output:

```
{
  "links": [ ... ],
  "name": "folders",
  "accept": "application/json application/vnd.sas.marketing.folder.summary+json
application/vnd.sas.collection+json",
  "start": 0,
  "count": 2,
  "items": [
    {
      "version": 1,
      "id": "A57MQUXZ-AA000E0V~Tree",
      "name": "Assets_for_Spring",
      "count": 0,
      "links": [ ... ],
      "modifiedTimeStamp": "2021-03-26T17:53:56Z",
      "modifiedBy": "sasUser@sas.com",
      "folderPath": "All Assets/Assets_for_Spring"
    },
    {
      "version": 1,
      "id": "A62MQBNZ-AB000E0V~Tree",
      "name": "Assets_for_Summer",
      "count": 0,
      "links": [ ... ],
      "modifiedTimeStamp": "2021-03-26T17:54:56Z",
      "modifiedBy": "sasUser@sas.com",
      "folderPath": "All Assets/Assets_for_Summer"
    }
  ],
  "limit": 50,
  "version": 2
}
```

- 3 Retrieve the list of digital assets that are in the folder. Send a GET call to the `/marketingDigitalAssets/digitalAssets` endpoint and set the folder's ID as the parent parameter. For example:

GET https://<external_gateway_host>/marketingDigitalAssets/digitalAssets?
parent=A57MQUXZ-AA000E0V~Tree

This is an example of a successful response when the sample folder contains one asset:

```
{
  "links": [ ... ],
  "name": "digitalAssets",
  "accept": "application/json application/vnd.sas.marketing.asset.summary+json
application/vnd.sas.collection+json",
  "start": 0,
  "count": 1,
  "items": [
    {
      "createdOn": "2020-08-27T17:44:45Z",
      "createdBy": "sasUser@sas.com",
      "createdByName": "sasUser@sas.com",
      "modifiedOn": "2020-08-27T17:44:45Z",
      "modifiedBy": "sasUser@sas.com",
      "modifiedByName": "sasUser@sas.com",
      "id": "89ba6e98-a3ed-4b4b-b25b-0e5255c49493",
      "name": "Spring_picture",
      "description": "This is a picture of day in Spring.",
      "deleteFlag": false,
      "assetIdentityPermissions": [],
      "attributeGroups": [],
      "version": 0,
      "revision": 1,
      "revisionNumber": 0,
      "revisionGuid": "b62e8f81-7ec6-4822-87ce-1f294c7dbc46",
      "currentRevision": true,
      "parentFolderPath": "All Assets/Assets_for_Spring",
      "parentFolderId": "A57MQUXZ-AA000E0V~Tree",
      "expired": false,
      "renditionSummary": [
        {
          "createdOn": "2020-08-27T17:44:46Z",
          "modifiedOn": "2020-08-27T17:44:49Z",
          "modifiedBy": "sasUser@sas.com",
          "version": 0,
          "name": "Thumbnail",
          "localizedName": "Thumbnail",
          "guid": "0a3cd3a1-87b0-4f86-ac40-164e8a522e94",
          "renditionTypeCd": "THM",
          "fileName": "Spring_picture_Thumbnail.jpeg",
          "fileFormat": "image/jpg",
          "fileSize": 4079,
          "dpi": 300.0,
          "dotsPerInch": 0.0,
          "height": 167,
          "width": 250,
          "depth": 24,
          "downloadCount": 0,
          "mediaTypeVal": "Image",
          "formatTypeVal": "JPG",
          "links": []
        }
      ],
    }
  ],
}
```

```

    {
      "createdOn": "2020-08-27T17:44:46Z",
      "modifiedOn": "2020-08-27T17:44:59Z",
      "modifiedBy": "sasUser@sas.com",
      "version": 0,
      "name": "Preview",
      "localizedName": "Preview",
      "guid": "460ed18d-797f-4308-84a5-b9be8d57fc21",
      "renditionTypeCd": "PRV",
      "fileName": "Spring_picture_Preview.jpeg",
      "fileFormat": "image/jpg",
      "fileSize": 16382,
      "dpi": 300.0,
      "dotsPerInch": 0.0,
      "height": 400,
      "width": 600,
      "depth": 24,
      "downloadCount": 0,
      "mediaTypeVal": "Image",
      "formatTypeVal": "JPG",
      "links": []
    },
    {
      "createdOn": "2020-08-27T17:44:46Z",
      "createdBy": "sasUser@sas.com",
      "modifiedOn": "2020-08-27T17:44:47Z",
      "modifiedBy": "sasUser@sas.com",
      "version": 0,
      "name": "Original",
      "localizedName": "Original",
      "guid": "ee85b219-7b51-474e-a9b2-200fe8ea0d48",
      "renditionTypeCd": "ORG",
      "fileName": "Spring_picture1.jpeg",
      "fileFormat": "image/jpeg",
      "fileSize": 75481,
      "dpi": 300.0,
      "dotsPerInch": 0.0,
      "height": 853,
      "width": 1280,
      "depth": 24,
      "downloadCount": 0,
      "mediaTypeVal": "Image",
      "formatTypeVal": "JPEG",
      "links": []
    }
  ],
  "links": []
},
"limit": 50,
"version": 2
}

```

Batch Jobs for Assets

About the REST API Endpoint for Batch Jobs

To initiate REST API operations for jobs that are related to batch processing of assets, use the endpoint `/assetJobs`. You can perform the following actions:

- initiate a batch processing job to upload and update multiple assets
- initiate a batch processing job to catalog assets and create new assets
- initiate a batch processing job to upload or update custom properties for multiple assets
- initiate a batch processing job to generate external sharing URLs for multiple digital assets
- view and manage batch processing jobs that are related to assets

The developer documentation for this REST API is available at [Digital Assets API](#).

Building the Base URL for the API Calls

To successfully use the Digital Assets API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external_gateway_host>/marketingDigitalAssets/digitalAssets
```

Example: Get a Collection of Digital Assets Jobs

This example demonstrates how to retrieve a collection of batch processing jobs that are related to uploading, updating, cataloging, and external sharing of multiple assets.

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 69 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiGci0.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve a collection of batch processing jobs that are related to assets:

a In the request header, set the Accept parameter to **application/vnd.sas.collection+json;version=1**.

b Use this GET call:

GET https://<external_gateway_host>/marketingDigitalAssets/assetJobs

3 This is an example of a successful response:

```
{
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "https://<external_gateway_host>/marketingDigitalAssets/assetJobs?start=0&limit=50",
      "uri": "/assetJobs?start=0&limit=50",
      "type": "application/vnd.sas.collection",
      "itemType": "application/vnd.sas.marketing.job"
    },
    {
      "method": "POST",
      "rel": "createUploadJob",
      "href": "https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/batchUpload",
      "uri": "/digitalAssets/batchUpload",
      "type": "application/json"
    },
    {
      "method": "POST",
      "rel": "createCatalogJob",
      "href": "https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/batchCatalog",
      "uri": "/digitalAssets/batchCatalog",
      "type": "application/json"
    },
    {
      "method": "POST",
      "rel": "createStandardAttributeJob",
      "href": "https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/batchStandardAttributeUpdate",
      "uri": "/digitalAssets/batchStandardAttributeUpdate",
      "type": "application/json"
    },
    {
      "method": "POST",
      "rel": "createCustomAttributeJob",
      "href": "https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/batchCustomAttributeUpdate",
      "uri": "/digitalAssets/batchCustomAttributeUpdate",
      "type": "application/json"
    },
    {
      "method": "POST",

```

```

        "rel": "createExternalizationJob",
        "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/
batchExternalization",
        "uri": "/digitalAssets/batchExternalization",
        "type": "application/json"
    }
],
    "name": "Jobs for Digital Assets",
    "accept": "application/json application/vnd.sas.marketing.job+json application/
vnd.sas.collection+json",
    "start": 0,
    "count": 2,
    "items": [
        {
            "version": 1,
            "id": "a3d92c1a-9c90-408e-9e2f-432457096d6a",
            "name": "BatchUpload2019-07-01T08:43:55Z",
            "localizedName": "BatchUpload2019-07-01T08:43:55Z",
            "createdBy": "sasuser",
            "creationTimeStamp": "2019-07-21T10:12:18Z",
            "modifiedBy": "sasuser",
            "modifiedTimeStamp": "2019-07-21T10:12:18Z",
            "totalItems": 2,
            "application": "ASSET",
            "type": "batchAssetUpload_micro",
            "state": "succeeded",
            "parallelJobSteps": false,
            "steps": [
                {
                    "version": 1,
                    "name": "asset upload",
                    "type": "assetUpload",
                    "state": "succeeded",
                    "asynchronous": false,
                    "creationTimeStamp": "2019-07-21T10:12:18Z",
                    "errorMessages": [],
                    "warningMessages": [
                        "No preview or thumbnail rendition generated for bannerartwork.cmake,
file type or content type not supported."
                    ]
                }
            ]
        }
    ],
    "links": [
        {
            "method": "GET",
            "rel": "self",
            "href": "https://<external gateway host>/marketingDigitalAssets/assetJobs/
a3d92c1a-9c90-408e-9e2f-432457096d6a",
            "uri": "/assetJobs/a3d92c1a-9c90-408e-9e2f-432457096d6a",
            "type": "application/vnd.sas.marketing.job"
        },
        {
            "method": "DELETE",
            "rel": "delete",
            "href": "https://<external gateway host>/marketingDigitalAssets/assetJobs/
a3d92c1a-9c90-408e-9e2f-432457096d6a",

```

```

        "uri": "/assetJobs/a3d92c1a-9c90-408e-9e2f-432457096d6a",
        "type": "application/vnd.sas.marketing.job"
    }
  ]
}

```

Asset Renditions

About the REST API Endpoint for Asset Renditions

To retrieve or update the renditions for the specified digital asset through the REST API, use the endpoint `/digitalAssets`. You must also specify the rendition ID and the digital asset ID. You can perform the following actions:

- retrieve a rendition or a collection of renditions for an asset
- create, update, and delete renditions
- retrieve the preview or the content of a rendition based on the specified digital asset ID, rendition ID, and revision number

The developer documentation for this REST API is available at [Digital Assets API](#).

Building the Base URL for the API Calls

To successfully use the Digital Assets API, start with the correct URL domain.

1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingDigitalAssets/digitalAssets
```

Example: Update a Rendition

This example updates the specified properties of a rendition and the revision based on the representation in the request body.

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 70 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Use a PATCH operation to update the specified rendition based on the specified digital asset ID and revision number in the request body:

PATCH https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/{digitalAssetId}/revisions/{revisionNumber}/renditions/{renditionId}

This is an example of a request body:

```
{
  "operations": [
    {
      "op": "replace",
      "path": "/name",
      "value": "New Custom Rendition"
    }
  ]
}
```

- 3 This is an example of a successful response:

```
{
  "version": 1,
  "id": "6e914138-c6c1-419e-97a9-3765db429526",
  "name": "New Custom Rendition",
  "localizedName": "Original",
  "createdBy": "sasuser",
  "creationTimeStamp": "2019-07-25T12:45:49Z",
  "modifiedBy": "rdt014",
  "modifiedTimeStamp": "2019-07-26T08:03:19Z",
  "parentAssetId": "80b44d55-41ca-4bac-8843-2e01e151b9dd",
  "renditionTypeCode": "org",
  "generationTypeCode": "usr",
  "fileName": "EmailBanner_Original.jpg",
  "fileFormat": "image/jpeg",
  "fileSize": 158112,
  "dotsPerInch": 0,
  "height": 1200,
  "width": 1920,
  "depth": 24,
  "duration": 0,
  "downloadCount": 0,
  "fileMediaType": "Image",
  "fileFormatType": "JPEG",
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748",

```

```

      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748",
      "type": "application/vnd.sas.marketing.rendition"
    },
    {
      "method": "PATCH",
      "rel": "patch",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748",
      "type": "application/vnd.sas.marketing.rendition"
    },
    {
      "method": "DELETE",
      "rel": "delete",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748",
      "type": "application/vnd.sas.marketing.rendition"
    },
    {
      "method": "GET",
      "rel": "parent",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2",
      "type": "application/vnd.sas.marketing.asset"
    },
    {
      "method": "GET",
      "rel": "getRenditions",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions",
      "type": "application/vnd.sas.collection"
    },
    {
      "method": "GET",
      "rel": "download",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748/download",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-e8e3-631e-19c1-5987fd641748/download",
      "type": "image/jpg"
    },
    {
      "method": "GET",
      "rel": "getContent",

```

```

      "href": "https://<external_gateway_host>/marketingDigitalAssets/digitalAssets/
80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/renditions/8g136350-
e8e3-631e-19c1-5987fd641748/content",
      "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2/
renditions/8g136350-e8e3-631e-19c1-5987fd641748/content",
      "type": "image/jpg"
    }
  ]
}

```

Asset Revisions

About the REST API Endpoint for Asset Revisions

To retrieve the revisions for the specified digital asset through the REST API, use the endpoint `/digitalAssets`. You must also specify the digital asset ID. You can perform the following actions:

- retrieve a revision of an asset based on the representation in the request body, or based on an existing revision
- retrieve a collection of revisions for an asset
- create or delete a revision of an asset

The developer documentation for this REST API is available at [Digital Assets API](#).

Building the Base URL for the API Calls

To successfully use the Digital Assets API, start with the correct URL domain.

1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external_gateway_host>/marketingDigitalAssets/digitalAssets
```

Example: Get the Revisions for a Digital Asset

This example retrieves all the revisions for a specified digital asset.

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 71 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the revisions for the digital asset

GET https://<external gateway host>/marketingDigitalAssets/digitalAssets/{digitalAssetId}/revisions

3 This is an example of a successful response:

```
{
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/{digitalAssetId}/revisions?start=0&limit;=50",
      "uri": "/digitalAssets/{digitalAssetId}/revisions?start=0&limit;=50",
      "type": "application/vnd.sas.collection",
      "itemType": "application/vnd.sas.marketing.revision"
    },
    {
      "method": "POST",
      "rel": "create",
      "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/{digitalAssetId}/revisions",
      "uri": "/digitalAssets/{digitalAssetId}/revisions",
      "type": "application/vnd.sas.marketing.revision"
    }
  ],
  "name": "Revisions",
  "accept": "application/json application/vnd.sas.marketing.revision+json application/vnd.sas.collection+json",
  "start": 0,
  "count": 2,
  "items": [
    {
      "version": 1,
      "id": "afce14a9-1b64-46dd-9992-02573d2acdd8",
      "comment": "First revision for banner",
      "currentRevision": false,
      "revisionNumber": 1,
      "modifiedBy": "sasuser",
      "modifiedTimeStamp": "2019-07-23T02:36:48Z",
      "parentAssetId": "80b44d55-41ca-4bac-8843-2e01e151b9dd",
      "links": [
        {
          "method": "GET",
          "rel": "self",
          "href": "https://<external gateway host>/marketingDigitalAssets/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/afce14a9-1b64-46dd-9992-02573d2acdd8",
```

```

        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/
revisions/afce14a9-1b64-46dd-9992-02573d2acdd8",
        "type": "application/vnd.sas.marketing.revision"
    },
    {
        "method": "GET",
        "rel": "parent",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/1",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/
revisions/1",
        "type": "application/vnd.sas.marketing.asset"
    },
    {
        "method": "GET",
        "rel": "getRevisions",
        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/
revisions",
        "type": "application/vnd.sas.collection"
    }
]
},
{
    "version": 1,
    "id": "bedf25b0-2c75-56ee-0003-13684e3bdee9",
    "comment": "New revision for banner",
    "currentRevision": true,
    "revisionNumber": 2,
    "modifiedBy": "sasuser",
    "modifiedTimeStamp": "2019-07-25T12:45:49Z",
    "parentAssetId": "80b44d55-41ca-4bac-8843-2e01e151b9dd",
    "links": [
        {
            "method": "GET",
            "rel": "self",
            "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/
bedf25b0-2c75-56ee-0003-13684e3bdee9",
            "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/
revisions/bedf25b0-2c75-56ee-0003-13684e3bdee9",
            "type": "application/vnd.sas.marketing.revision"
        },
        {
            "method": "GET",
            "rel": "parent",
            "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions/2",
            "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/
revisions/2",
            "type": "application/vnd.sas.marketing.asset"
        },
        {
            "method": "GET",
            "rel": "getRevisions",

```

```

        "href": "https://<external gateway host>/marketingDigitalAssets/
digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/revisions",
        "uri": "/digitalAssets/80b44d55-41ca-4bac-8843-2e01e151b9dd/
revisions",
        "type": "application/vnd.sas.collection"
    }
}
]
}
}

```

Marketing Administration API

About the Marketing Administration API

IMPORTANT The developer documentation is available at [Marketing Administration API](#).

Introduction to the Marketing Administration API

The Marketing Administration API enables you to perform administration actions for SAS Customer Intelligence 360:

- review audit records for actions that occurred on items in SAS Customer Intelligence 360
- manage access tokens that are used by access points and users to authenticate with REST APIs

Endpoints in the Marketing Administration API

The root of the Marketing Administration API is `/marketingAdmin/`. Combine the root with one of these endpoints:

- `/auditRecords` can be used to download messages that are collected by the audit service.
- `/tokenManagement` can be used to manage access tokens that are associated with access points.

For examples of using these endpoints, see [“Examples for the Marketing Administration API” on page 179](#).

Prerequisites to Use the Marketing Administration API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see [“Setting Up Authorization” in SAS Customer Intelligence 360: Administration Guide](#).

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.
 When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable *<external gateway host>* in the documentation.
 - b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.
- 2 Generate the temporary JWT with your static JWT or the client_jwt value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the Marketing Administration API

Audit Records

Introduction to the REST API for Audit Records

You can use this REST API to download audit records from SAS Customer Intelligence 360. Audit records contain information about actions that are performed by users in the user interface or REST API.

Here are some details about the audit records that are available:

- SAS Customer Intelligence 360 retains up to 558 days of audit records.
- You can download 31 days of records through one request to the Marketing Administration API. To download more than 31 days, you must send multiple requests.
- IP addresses and user agents are recorded only for sign-in and sign out events. However, every record contains a session ID. Therefore, you can use the session ID value as a cross-reference value to determine the IP address or user agent for other actions.
- The start date and end date are rounded to the nearest hour interval. The values for minutes and seconds are ignored if they are present in the timestamp values. For example, if you request data from 8:30AM to 9:30AM, you receive records from 8:00AM to 8:59AM.

Note: SAS Customer Intelligence 360 will not accept any requests where the time range is less than an hour.

- Audit records can be downloaded for a specific hour after a 10-minute wait for processing. For example, to download records from 8:00AM to 9:00AM, you can request the records at 9:10AM.

- If you do not provide date parameters, you receive the most recently processed hour of audit records.
- If you provide only one date parameter (dataRangeStartTimeStamp or dataRangeEndTimeStamp), you receive only an hour of audit records that is based on the parameter that you specified.

The download files use the Apache Parquet format. For more information, see “[Schema for Audit Records](#)” in *SAS Customer Intelligence 360: Administration Guide*.

The developer documentation is available at [Marketing Administration API](#).

Building the Base URL for the API Calls

To successfully use the REST API, start with the correct URL domain.

- 1 Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- 2 Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external gateway host>/<endpoint>`

- 3 Use this base URL with the API paths to make calls. For example:

GET `https://<external gateway host>/marketingAdmin/auditRecords`

Example: Download Audit Records

To download audit records:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 72 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer <code>eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v</code>

- 2 Send a GET call to the `/auditRecords` endpoint:

GET `https://<external gateway host>/marketingAdmin/auditRecords?
dataRangeStartTimeStamp=2023-06-14T01:00Z&
dataRangeEndTimeStamp=2023-06-14T02:00Z`

This GET call can use these query parameters in the URL:

Table 73 Parameters for Audit Records

Name	Required or Optional	Data Type	Description
dataRangeStartTimeStamp	Optional	Timestamp	The start timestamp to use. The timestamp must be in UTC format, such as: <pre>2023-06-13T01:00Z</pre> The start date and end date are rounded to the nearest hour interval. The values for minutes and seconds are ignored if they are present in the timestamp values. For example, if you request data from 8:30AM to 9:30AM, you receive records from 8:00AM to 8:59AM. If you provide only one date parameter (dataRangeStartTimeStamp or dataRangeEndTimeStamp), you receive only an hour of audit records that is based on the parameter that you specified. Note: SAS Customer Intelligence 360 will not accept any requests where the time range is less than an hour.
dataRangeEndTimeStamp	Optional	Timestamp	The end timestamp to use. The timestamp must be in UTC format, such as: <pre>2023-06-14T02:00Z</pre> Note: The value must be within 31 days of the value for the dataRangeStartTimeStamp parameter. To download data for a longer period, you must send multiple requests. You receive a 500 error if you request more data than is available or the start date is after the end date.

In the response, the items property contains an array with one or more URLs to the files that contain the audit records. This is an example of a successful response:

```
{
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "http://<external gateway host>/marketingAdmin/auditRecords?
dataRangeEndTimeStamp=2023-06-14T02:00Z
      &dataRangeStartTimeStamp=2023-06-13T01:00Z",
      "uri": "/marketingAdmin/auditRecords?
dataRangeEndTimeStamp=2023-06-14T02:00Z&dataRangeStartTimeStamp=2023-06-13T01:00Z"
    }
  ],
}
```

```

    "name": "items",
    "accept": "application/vnd.sas.collection+json",
    "start": 0,
    "count": 1,
    "items": [
      "https://<download URL>"
    ],
    "limit": 1,
    "version": 2
  }

```

Token Management

Introduction to the REST API for Token Management

This REST API enables you to manage access tokens that are associated with access points in SAS Customer Intelligence 360. Users can make REST calls with the access tokens to interact with secure data.

This REST API enables you to complete these tasks:

- list the access points
- create an access token for an access point
- get a summary of existing access tokens
- get the details for a single access token
- update the status of an access token
- delete an access token

The developer documentation is available at [Marketing Administration API](#).

Building the Base URL for the API Calls

To successfully use the REST API, start with the correct URL domain.

- 1 Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- 2 Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable *<external gateway host>* in the documentation.

An example might use the following URL for an endpoint:

`https://<external gateway host>/<endpoint>`

- 3 Use this base URL with the API paths to make calls. For example:

`GET https://<external gateway host>/marketingAdmin/tokenManagement/accessPoints`

Example: List Access Points

To list the access points that are associated with a tenant:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 74 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a GET call to the /tokenManagement/accessPoints/ endpoint:

GET https://<external gateway host>/marketingAdmin/tokenManagement/accessPoints

This is an example of a successful response:

```
{
  "count": 2,
  "items": [
    {
      "id": "a49d694e-a9e2-4104-8cbc-6e59c690d051",
      "name": "ap1",
      "status": "active"
    },
    {
      "id": "b76d695a-b3f1-4104-8cbc-6e59c690d051",
      "name": "ap2",
      "status": "active"
    }
  ]
}
```

Example: Disable an Access Token

To disable an access token that already exists:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 75 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a GET call to the /tokenManagement/accessPoints/ endpoint:

GET `https://<external_gateway_host>/marketingAdmin/tokenManagement/accessPoints`

This is an example of a successful response:

```
{
  "count": 2,
  "items": [
    {
      "id": "a49d694e-a9e2-4104-8cbc-6e59c690d051",
      "name": "access_pt_1",
      "status": "active"
    },
    {
      "id": "b76d695a-b3f1-4104-8cbc-6e59c690d051",
      "name": "access_pt_2",
      "status": "active"
    }
  ]
}
```

Take note of the ID of the access point that contains the token you want to disable.

- 3 Get a list of existing access tokens. Send a GET call to the following URL:

GET `https://<external_gateway_host>/marketingAdmin/tokenManagement/accessPoints/a49d694e-a9e2-4104-8cbc-6e59c690d051/tokens`

This is an example of a successful response:

```
{
  "count": 2,
  "items": [
    {
      "token_id": "b506e2bc-8269-4d22-8817-9a0986710e49",
      "created": "2023-06-06T20:48:15Z",
      "last_updated": "2023-06-08T14:22:35Z",
      "last_used": "",
      "expires_at": "2023-06-06T21:41:35Z",
      "modified_by": "API-infr1910-foobar",
      "enabled": true
    },
    {
      "token_id": "f326e8ca-4578-3e18-9191-0b3853692d13",
      "created": "2023-06-06T20:52:00Z",
      "last_updated": "2023-06-06T20:52:00Z",
      "last_used": "",
      "expires_at": "2023-06-12T21:41:35Z",
      "modified_by": "API-infr1910-foobar",
      "enabled": false
    }
  ]
}
```

Take note of the value for the `token_id` attribute for the access token that you want to disable.

- 4 Disable the access token. Send a PUT call to the following URL:

PUT `https://<external_gateway_host>/marketingAdmin/tokenManagement/accessPoints/a49d694e-a9e2-4104-8cbc-6e59c690d051/tokens/b506e2bc-8269-4d22-8817-9a0986710e49/enabled`

Use this JSON code for the body of the request:

```
{  
  "enabled": false  
}
```

You receive a 204 status when the call is successful.

Marketing Audience API

About the Marketing Audience API

IMPORTANT The developer documentation for this API is available at [Marketing Audience API](#).

Introduction to the Marketing Audience API

The Marketing Audience REST API provides a way to create and manage [audiences](#) in SAS Customer Intelligence 360.

Note: In order to access audiences, you need a license for SAS 360 Audiences.

If your data comes from Snowflake, Google BigQuery, or Amazon Redshift create the audience from the [Audiences page](#). You can use the API to perform these actions on audiences that use cloud data:

- retrieve information about the audience
- retrieve a list of audiences
- delete an audience

To create an audience from a data source that SAS Customer Intelligence 360 does not have a direct connection to, create an audience definition either with the [Marketing Audience API](#) or from the [Audience Definitions page](#). When you use the API to create an audience, an audience definition and the corresponding audience are created. After creating the audience definition, use the API to import a CSV file that contains the audience data.

These are the actions that you can perform on audiences created from audience definitions:

- create an audience
- retrieve information about the audience
- import data into an audience
- retrieve information about an import request
- update an audience, such as adding new columns or renaming columns that are not currently in use

- delete an audience
- get a list of all audiences

To see the list of audience definitions in the user interface, navigate to  **Administration** ⇒ **Data and Integrations** ⇒ **Audience Definitions**.

Endpoints in the Marketing Audience API

The root of the Marketing Audience API is `/marketingAudience`. Combine the root with the `/audiences` endpoint to manage the audiences in the application.

For examples of how to use this endpoint, see [“Example: Create an Audience” on page 189](#).

Prerequisites to Use the Marketing Audience API

To use this API, you must authenticate with a temporary JSON Web Token (JWT). To generate a temporary JWT:

- 1 Make sure that you have completed these prerequisites:
 - a [Generate a static JWT](#) or have the `client_jwt` value from an access token.
 - b [Create a definition for an API credential](#). Make note of the user name and secret.
- 2 In a REST client (such as Postman), send a POST request to the `/token` endpoint.
 - a Specify the static JWT as the value for the Authorization header.

.....
Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BR v_nD6v

- b Set the value of the Content-type header to `application/x-www-form-urlencoded`.
- c Add the following parameters in the request body:
 - Add the **grant_type** parameter and set the value to **password**.
 - Add the **username** parameter and set the value to the API user ID. The API user ID was specified when the API credentials were defined.
 - Add the **password** parameter and set the value to the secret that was generated when the API credentials were defined.

- (Optional) Add the **expiration** parameter and set the length of time (in seconds) for which the temporary token is valid. By default, the token is valid for 12 hours (43200 seconds).

For example, if you create API credentials with an API user ID of "API-userName-example", and the generated secret is "3DHS8C3NT1RC5XKV", your request body contains these parameters:

Parameter	Value
grant_type	password
username	API-userName-example
password	3DHS8C3NT1RC5XKV

- d Send a POST call to the REST API:

POST `https://<external_gateway_host>/token`

TIP To find the value of the `<external_gateway_host>` variable, navigate to  **Administration** ⇒ **Data and Integrations**, and click **Access Points**. The value is in the URL domain of the **External host address** field.

This is an example of a successful response:

```
{
  "access_token": "BFmuv+i+/b8Ef1cf4ufm3R6ZmH3pFx168Ez516iB/91i",
  "token_type": "bearer",
  "expires_in": 43200
}
```

Note: By default, the temporary JWT is valid for 12 hours.

Use the value of the `access_token` attribute as the bearer token to authenticate with secure APIs.

Marketing Audience API Statuses

This is a list of the terminal statuses for the Marketing Audience API.

Table 76 *Statuses*

Status	Description
CANCELED	The audience processing did not complete because it was canceled by the user.
COMPLETE	The audience processing completed successfully.
CONNECTION_INACTIVE	The audience processing failed because the connection that is used by the audience is inactive.
CONNECTION_NOT_FOUND	The audience processing failed because the connection that is used by the audience cannot be found.
CONNECTION_TABLE_COLUMNS_MISSING	The audience processing failed because one or more columns that are used by the audience are missing from an underlying table.
CONNECTION_TABLE_MISSING	The audience processing failed because an underlying table in the connection is missing.
DUPLICATE_IDENTITY_VALUES	The audience processing failed because the identity column contains duplicate identity values.
EXPIRED, LICENSE_FAILED	The audience processing failed because the SAS 360 Audiences license has expired.
FILE_COLUMN_DATA_TYPES_NOT_MATCH	The audience processing failed because the data type for a column in the audience definition does not match the corresponding column data type in the uploaded file.
FILE_INVALID_FORMAT	The audience processing failed because the uploaded file has an invalid file format.
FILE_NOT_FOUND	The audience processing failed because the uploaded file cannot be found.
FILE_PROCESSING_FAILED	The audience execution failed because the uploaded file could not be processed.
GDPR_PROCESSED	The GDPR process completed successfully.
HARD_ROW_LIMIT_EXCEEDED	The audience processing failed because the audience size exceeds the audience row limit.
IDENTITY_COLUMN_NOT_UNIQUE	The audience execution failed because its configured identity column was not unique.

Status	Description
IDENTITY_RESOLUTION_FAILED	The audience processing failed due to an issue with the internal identity resolution process.
NO_DATA_FOUND	The audience processing failed because the uploaded file did not contain data or it was not possible to parse the data correctly.
QUERY_FAILED	The audience processing failed because the query to the underlying source failed.
SEGMENT_FAILED	The audience processing failed because segment data cannot be retrieved.
SEGMENT_MAP_CANNOT_BE_RUN	The audience processing failed because the underlying segment map cannot be run.
SEGMENT_MAP_NOT_ACTIVE	The audience processing failed because the underlying segment map is not active.
SEGMENT_MAP_NOT_FOUND	The audience processing failed because the underlying segment map cannot be found.
SOFT_ROW_LIMIT_EXCEEDED	The audience processing failed because the audience size exceeded the audience row limit.
SUCCESS	The audience process completed successfully.
TENANT_CMK_PAUSE	The audience processing failed because the tenant's customer-managed key is paused.
TENANT_CMK_REVOKED	The audience processing failed because the tenant's customer-managed key was revoked.
UNKNOWN_EXCEPTION	The audience processing failed due to an unknown exception.

Example: Create an Audience

Overview of Creating an Audience

These are the high-level steps required to create an audience from data that the application does not have a direct connection to.

- 1 Create the audience definition. This step requires that you enter the schema for the audience definition in the API call. The corresponding audience with the same name is automatically created during this step.
- 2 Use the API to get a temporary location where you can upload a CSV file that contains the audience data.
- 3 Upload the CSV file to the temporary location so that it can be imported into the application.
- 4 Process the audience data so that the audience becomes active and can be used throughout SAS Customer Intelligence 360.

Parameters Used in Example

This example illustrates how to create an audience with three columns: Customer ID, ZIP code, and Birthday. These are the audience parameters that are used in the example.

Table 77 *Parameters Used in Example Audience*

Name	Description
name	Name of the audience.
description	Describes the audience.
expiration	The number of days before the audience expires. This value determines how long the data is available in the application and is based on UTC. The maximum value allowed is 180 and the expiration countdown starts when the audience is run. The audience data expiration is calculated using the most recent date that the audience data was successfully processed.
identityType	The value that is used as the identity type.
identityColumnName	The column that contains identity values such as customer ID, login ID, and so on.
emailColumnName	The column that contains the email addresses that are used as the default contact information. Note: This cannot be the same column that was specified for the identityColumnName parameter.

These are the parameters that the example uses for each column, or data item.

Table 78 *Parameters Used for Example Columns*

Name	Description
columnNumber	The column index.

Name	Description
name	Customizes the column names by replacing the default text with more meaningful names. If these are not customized, the column names assigned have the default syntax of "col1", "col2", and so on. These names are displayed in the content of a task, such as the HTML of an email, in this format: %AUDIENCE.TARGET.<column_name>%. If you are viewing an audience definition in Administration ⇒ Data and Integrations ⇒ Audience Definitions, this parameter is called "Column Name".
label	Customizes the label if the column names in your data are not in a format that is easily understood by others. These are the names that are displayed when a data item is selected. For example, the label is displayed when a user selects the data item as a merge tag in a task, a display condition in email content, or as a split attribute in a scheduled journey. If you are viewing an audience definition in Administration ⇒ Data and Integrations ⇒ Audience Definitions, this parameter is called "Displayed Column Name".
dataType	The data type to use for the column.
identifiableInformation	Denotes that data can be used to identify someone, so that the data item can be categorized as personally identifiable information (PII). This property was previously named "Enable for Event Attribute Tracking". If you created an audience definition with the old name, this previous name for the property is present when you download the data with the REST API. However, if the audience definition is opened from the Audiences page, you see the "Identifiable Information" name instead. Note: Be aware of these notes when you set this value: ■ If both the <code>usedForTracking</code> property and the <code>identifiableInformation</code> property are provided for a column, the values must be opposites. The <code>identifiableInformation</code> property is used to specify which columns contain data that can be used to identify someone, so that the data item can be categorized as personally identifiable information (PII). If only one property is provided, the REST API infers the value of the other property. If both properties are omitted, the <code>usedForTracking</code> property is set to <code>false</code> and the <code>identifiableInformation</code> property is set to <code>true</code>. ■ When the <code>usedForTracking</code> property is <code>true</code> or the <code>identifiableInformation</code> property is <code>false</code>, these data attributes can appear in places in the Unified Data Model (UDM) tables where they cannot be deleted. Do not set the <code>usedForTracking</code> property to <code>true</code> or the <code>identifiableInformation</code> property to <code>false</code> on columns that contain sensitive data.

Steps to Create Example Audience Definition

Follow these steps to create the example audience definition:

- 1 Ensure that you can access the API. For more information, see [“Prerequisites to Use the Marketing Audience API” on page 186](#).

Generate a new temporary JWT token if it is expired. To successfully use the API endpoints, make sure to add the valid JWT to the Authorization attribute of the Header. The header should be similar to this example:

Table 79 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Create the audience definition and audience. This is an example of the POST call:

POST `https://<external_gateway_host>/marketingAudience/audiences`

In the body of the call, enter the JSON code for the audience definition. Here is an example of the JSON for an audience definition with three columns:

```
{
  "name": "Sample Definition",
  "description": "A definition created via the API",
  "expiration": 10,
  "identityType": "customer_id",
  "identityColumnName": "Customer ID",
  "emailColumnName": "Zip Code",
  "dataItems": [
    {
      "columnNumber": 1,
      "name": "customer_id",
      "label": "Customer ID",
      "dataType": "character",
      "identifiableInformation": "false"
    },
    {
      "columnNumber": 2,
      "name": "zip_code",
      "label": "Zip Code",
      "dataType": "character",
      "identifiableInformation": "true"
    },
    {
      "columnNumber": 3,
      "name": "birthday",
      "label": "Birthday",
      "dataType": "date",
      "identifiableInformation": "false"
    }
  ]
}
```

At this point, the audience definition is active and the audience is inactive. You can see the audience definition that was created by navigating to  **Administration** ⇒ **Data and Integrations**

⇒ **Audience Definitions.** You can see the audience that was created by navigating to **Audiences** in the left navigation.

- 3 To get a URL to upload your CSV, send a POST call (with no JSON body) to this URL:

POST `https://<external_gateway_host>/marketingAudience/audiences/fileTransferLocation`

In the response, take note of the `signedURL` property:

```
{
  "version": 1,
  "signedURL": "<temporary URL>",
  "httpMethod": "POST",
  ...
}
```

Note: Temporary URLs can be accessed for about 15 minutes before they time out. You can use temporary URLs more than once, but you can always request a new URL if you do not start your upload in the 15-minute window.

- 4 Use curl or a similar application to upload the import file to the temporary URL. This is an example of a curl command to upload a file from your local machine:

```
curl -v --upload-file "c:\temp\myfile.csv" "https://<temporary_URL>"
```

At this point, the audience data is uploaded. However, the audience remains inactive until the audience data is processed.

- 5 Activate the audience by processing the data using a PUT call like this example:

PUT `https://<external_gateway_host>/marketingAudience/audiences/<audienceID>/data`

TIP Use a GET call to the `marketingAudience/audiences` endpoint to retrieve a list of audiences and their IDs.

In the body of the call, enter the JSON code containing the required parameters. Here is an example of the JSON:

```
{
  "audienceId": "asfsaf-wer234-ce4asfd234234",
  "name": "My New Audience Definition",
  "fileLocation": "https://<server>/transfers",
  "headerRowIncluded": true
}
```

Table 80 Parameters for Audience Processing

Name	Required or Optional	Description
name	Optional	The name of the import.
audienceId	Required	The value of the audienceId attribute for the audience that you want to process.

Name	Required or Optional	Description
fileLocation	Required	The signed URL that you generated from <code>theMarketingAudience/audiences/fileTransferLocation</code> endpoint for the value of the <code>fileLocation</code> parameter.
headerRowIncluded	Required	Specifies whether the file contains a header row. Set the value to <code>true</code> if you want to skip processing on the first row in the data file.
ignoreDotConversion	Optional	By default, occurrences of a "." (a period that denotes missing information) are handled as null when importing the data. To override this behavior, set the value of the <code>ignoreDotConversion</code> parameter to <code>true</code> .

At this point, your audience should be active.

TIP To see the history of your audience or to get information about why the upload failed, send a GET call to the `audiences/<audience_id>/history` endpoint.

GET `https://<external_gateway_host>/audiences/<audience_id>/history`

Marketing Data API

About the Marketing Data API

IMPORTANT The developer documentation for this API is available at [Marketing Data API](#).

Introduction to the Marketing Data API

The Marketing Data API for SAS Customer Intelligence 360 enables you to manage data for customers and analytics. For example, you can use this REST API to perform these actions:

- fulfill GDPR requests
- import and update customer data
- control product recommendations

Endpoints in the Marketing Data API

The root of the Marketing Data API is `/marketingData`. Combine the root with one of these endpoints:

- `/analytic` contains operations to control product recommendations. For more information, see [“Analytics and Product Recommendations”](#).
- `/customerJobs` contains operations to fulfill GDPR requests. For more information, see [“GDPR Support” in SAS Customer Intelligence 360: Administration Guide](#).
- `/fileTransferLocation` provides a short-lived URL that you can use to upload files to SAS Customer Intelligence 360. For more information, see [File Transfer Location](#) in the developer documentation.
- `/importRequestJobs` contains operations to import data that is associated with an existing table.
- `/identityRecords` contains operations to retrieve information about the identities that are stored in SAS Customer Intelligence 360. For more information, see [“Introduction to the REST API for Identities”](#).
- `/tables` and `/tableJobs` contain the operations to retrieve information about customer tables and upload the corresponding data files. For more information, see [“Customer Tables”](#).

For examples of how to use these endpoints, see [“Examples for the Marketing Data API” on page 196](#).

Prerequisites to Use the Marketing Data API

If you do not already have a static JWT or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an access point so that you can generate a static JWT.
When you create the access point, make note of the **External gateway host** field on the Access Points page. This value is used as part of the URL for API calls that go through the external API gateway. Use this value when you see the variable `<external_gateway_host>` in the documentation.
- 2 (Optional) Create an API credential if you want to create a time-limited access token. This REST API does not require a temporary JWT, so you need to create an API credential only if you are completing the one-time setup to create an access token.
- 3 Make note of the value of the token that you need to use. The static JWT or access token must be specified in the Authorization header as a bearer token when you make calls to the REST API.

Examples for the Marketing Data API

Analytics and Product Recommendations

Introduction to the REST API for Analytics

To manage analytic information, you use the `/analytic` endpoint in the Marketing Data API (`/marketingData`). With this REST API, you can perform these actions:

- create a list of allowed products or disallowed products for recommendations
- view the status of existing analytic jobs

The developer documentation for this API is available at [Marketing Data API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Data API, start with the correct URL domain.

- 1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingData/analytic/transfers
```

Example: Creating a Recommendation Denylist or Allowlist

About Denylists and Allowlists for Recommendation Tasks You can define which products are available to recommendation tasks by uploading a CSV file that contains a list of products. That CSV file can be used as either an allowlist or a denylist.

Note: The CSV file is applied to all recommendation tasks.

A *recommendation allowlist* details which products are approved for use in recommendation tasks. After an allowlist is uploaded to SAS Customer Intelligence 360, recommendation tasks can recommend only the products that exist in the list.

A *recommendation denylist* details which products are excluded from use in recommendation tasks. After a denylist is uploaded to SAS Customer Intelligence 360, recommendation tasks cannot recommend any products that exist in the list.

For example, you might create a denylist that prevents the recommendation of any product that is not in your inventory. Alternatively, you might use a denylist to prevent winter-themed products from displaying during summer months.

TIP You can use the steps described here to create a script that automatically updates your imported list.

Process Overview To establish an allowlist or denylist for recommendations tasks, follow these steps:

- 1 Create and upload a data descriptor for recommendations using a POST call to the REST API. This call returns a temporary, signed URL where you upload your CSV file.
- 2 Upload the CSV file to the temporary URL.

After the CSV file is uploaded, recommendation tasks start using the denylist or allowlist when they deliver recommendations. User-centric tasks might take up to one day to begin using the imported list. Product-centric tasks might take up to one week to begin using the list.

Prerequisites In order to use a denylist or allowlist with recommendation tasks, you will need to upload some files to SAS Customer Intelligence 360 using the [Marketing Data REST API](#). Ensure that you satisfy these requirements before you begin:

- 1 Create an access point so that you can generate a static JWT.
When you create the access point, make note of the **External gateway host** field on the Access Points page. This value is used as part of the URL for API calls that go through the external API gateway. Use this value when you see the variable *<external_gateway_host>* in the documentation.
- 2 Make note of the value of the token that you need to use. The static JWT or access token must be specified in the Authorization header as a bearer token when you make calls to the REST API.
- 3 Create the CSV file that lists the products that are either allowlisted or denylisted.

Note: Note: Only one type of list is supported at a time – either an allowlist or a denylist. Take time to determine which type is more beneficial to you before you continue.

- 4 Create a data descriptor for recommendations. This descriptor specifies whether you are uploading an allowlist or a denylist and defines the structure of your CSV file.

Create the CSV File for Recommendations You must have a CSV file that contains the list of products that you want to denylist or allowlist. These are the requirements for the CSV file:

- The CSV file must contain a column named either `PRODUCT_ID` or `PRODUCT_SKU` to be valid. The file can contain both columns.
- Use UTF-8 encoding.
- Ensure that there are no empty data items.

Here is an example of a short CSV file:

```
PRODUCT_ID,PRODUCT_SKU,CATEGORY
1234,1234990,Sport
1235,1235993,Sport
1236,12365995,Baby
```

Create the Data Descriptor for Recommendations Data descriptors are JSON objects that describe the structure and the type of data items that are contained in a data set. In this case, your descriptor describes the structure of your CSV file.

- 1 Open a text editor and create a basic template. You can use text similar to this example:

```
{
  "listType": "",
  "columns": [
    {
      "name": "",
      "type": "",
      "length": ""
    },
    {
      "name": "",
      "type": "",
      "length": ""
    }
  ]
}
```

- 2 Define whether you are uploading an allowlist or a denylist by entering the appropriate value for the `listType` attribute. Valid values are **allowlist** or **denylist**.
- 3 Describe the structure of your CSV file by adding information in the `columns` attribute. Add an item in the columns list for each corresponding column in your CSV file. For example, if you have three columns in the CSV file, the columns attribute should have three children.

Add details about each of your columns using these attributes:

Table 81 CSV File Structure

Name	Type	Description
Name	String	The name of the column in your CSV file. One column must be named PRODUCT_ID or PRODUCT_SKU .
Type	String	SAS data type of your column. Valid values are CHARACTER or NUMBER .
Length	Integer	The length of the column field in your CSV file.

This example data descriptor is for a denylist:

```
{
  "listType": "denylist",
  "columns": [
    { "name": "PRODUCT_ID", "type": "CHARACTER", "length": 80 },
    { "name": "PRODUCT_SKU", "type": "CHARACTER", "length": 90 },
    { "name": "CATEGORY", "type": "CHARACTER", "length": 100 }
  ]
}
```

```
]
}
```

Upload the Data Descriptor for Recommendations

TIP The developer documentation for this REST API is available at [marketingData/analytic/transfers](#).

- 1 To authenticate with the external API gateway, specify the JWT as the value for the Authorization header in the REST API call.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 82 Using a JWT as the Authorization Header

Header Attribute	Header Value
Authorization	Bearer eyJ0eXAiOiOGci0.DAxMGFhODEJ9.BRv_nD6v

- 2 In the body of the REST call, enter the JSON code for the data descriptor.
- 3 Send a POST call to the REST API:

`https://<external_gateway_host>/marketingData/analytic/transfers`

TIP The response status is 201 when the descriptor is created successfully.

- 4 Notice the URL that is included in the response. This is a temporary location that you can use to upload the corresponding CSV file.

Note: Temporary URLs can be accessed for about 15 minutes before they expire. You can use temporary URLs more than once, but you can always request a new URL if you do not start your upload in the 15-minute window.

In addition to the URL, the response includes the `transferId` attribute. You can use this ID with a GET call to review the details of the transfer request (or see the temporary URL again):

`https://<external_gateway_host>/marketingData/analytic/transfers/<transferId>`

Upload CSV File for Recommendations Each time that a new file is uploaded, the existing list is replaced. Use curl or a similar application to upload the CSV file to the temporary URL. After your file is uploaded, note the value of the `transferId` attribute that is returned. This can be used to verify whether your file uploaded successfully.

This is an example of a curl command to upload a file from your local machine:

```
curl -v --upload-file "c:\temp\myfile.csv" "https://<temporary_URL>"
```

If the curl command is successful, the response includes the string “HTTP/1.1 200 OK”.

TIP For more information about curl, see <https://curl.se/> (external).

Validate Your Recommendation Upload You can validate that your file uploaded correctly by using one of these methods:

- Review the notifications in SAS Customer Intelligence 360. A notification appears after your upload succeeds.
- Send a GET call to the REST API using the `transferID` attribute that was returned after you uploaded the file.

`https://<external_gateway_host>/marketingData/analytic/transfers/<transferId>`

The response status is 200 when the file uploaded successfully. If the response status is 400, the file did not upload successfully.

Customer Tables

Introduction to the REST API for Customer Tables

To manage data tables that are stored in SAS Customer Intelligence 360 through a REST API, use endpoints in the Marketing Data API. With this REST API, you can perform these actions:

- create or modify definitions for customer tables in SAS Customer Intelligence 360
- create jobs to import customer data into existing tables
- download raw customer tables from the system

The developer documentation for this API is available at [Marketing Data API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Data API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external_gateway_host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

GET `https://<external_gateway_host>/marketingData/tables`

Example: Download a Customer Table

In this example, you download a customer table by retrieving its ID, and then creating an export job. To download a customer table:

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 83 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the list of tables:

GET https://<external gateway host>/marketingData/tables

In the response, make note of the table's ID. This is an example of a successful response:

```
{
  "links": [],
  "name": "items",
  "count": 4,
  "items": [
    {
      "version": 1,
      "name": "my_customer_table",
      "id": "26c9ac61-b642-491a-ab98-69608a789109",
      "links": [
        {
          "method": "GET",
          "rel": "self",
          "href": "https://<external gateway host>/marketingData/tables/26c9ac61-b642-491a-ab98-69608a789109",
          "uri": "/tables/26c9ac61-b642-491a-ab98-69608a789109",
          "type": "application/vnd.sas.marketing.table"
        },
        {
          "method": "DELETE",
          "rel": "delete",
          "href": "https://<external gateway host>/marketingData/tables/26c9ac61-b642-491a-ab98-69608a789109",
          "uri": "/tables/26c9ac61-b642-491a-ab98-69608a789109",
          "type": "application/vnd.sas.marketing.table"
        }
      ]
    },
    ...
  ],
  "version": 2
}
```

3 Send a POST call to the marketingData/tableJobs endpoint to start the export request.

POST https://<external gateway host>/marketingData/tableJobs

In the body of the request, set the jobType property and dataDescriptorId property like this example:

```
{
  "jobType": "TABLE_DOWNLOAD",
  "dataDescriptorId": "<table ID>"
}
```

The response contains short-lived URLs that you can use to download the corresponding table and the header file. For example, this could be the response that uses the table ID from the previous output sample:

```
{
  "jobType": "TABLE_DOWNLOAD",
  "version": 1,
  "dataDescriptorId": "26c9ac61-b642-491a-ab98-69608a789109",
  "headerRowIncluded": false,
  "downloadItemList": [
    {
      "path": "tableExport/2018071611/26c9ac61-b642-491a-ab98-69608a789109/20210403185203/26c9ac61-b642-491a-ab98-69608a789109.csv.part_00000",
      "url": "https://<server>/tableExport/2018071611/26c9ac61-b642-491a-ab98-69608a789109/20210403185203/26c9ac61-b642-491a-ab98-69608a789109.csv.part_00000?..."
    },
    {
      "path": "tableExport/2018071611/26c9ac61-b642-491a-ab98-69608a789109/20210403185203/header_26c9ac61-b642-491a-ab98-69608a789109.csv.part_00000",
      "url": "https://<server>/tableExport/2018071611/26c9ac61-b642-491a-ab98-69608a789109/20210403185203/header_26c9ac61-b642-491a-ab98-69608a789109.csv.part_00000?..."
    }
  ]
}
```

GDPR Requests and Customer Identities

IMPORTANT The developer documentation for this API is available at [Marketing Data API](#).

Introduction to the REST API for Identity Management

To use the REST API to manage identities and fulfill GDPR requests, use the `customerJobs` endpoint in the Marketing Data API (`/marketingData`). With this REST API, you can perform these actions:

- export all event data that is related to a customer's identity
- delete event data for a customer

Building the Base URL for the API Calls

To successfully use the Marketing Data API, start with the correct URL domain.

- 1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.

- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingData/customerJobs
```

Example: Delete Identity Data

To submit a request to delete identity data:

- 1 [Set up authorization for the REST API](#). For REST API calls, in most situations the access point type is going to be a General Access Point.

When you create the access point, make note of the value in the **External gateway host** field. Use this value when you see the variable `<external gateway host>` in the documentation.

- 2 Send a POST call to the customerJobs endpoint to delete data. The call is similar to this example:

```
POST https://<external gateway host>/marketingData/customerJobs
```

Note: Be sure to use the JWT as the authentication token when you submit any REST calls.

In the body of the call, define the delete request as a JSON object:

```
{
  "jobType": "GDPR_DELETE",
  "identityType": "subject_id",
  "identityList": [
    "12343455"
  ]
}
```

The delete request contains these members:

Table 84 Members of the Customer Delete Request

Name	Required or Optional	Type	Description
jobType	Required	String	The job type. Set this value to GDPR_DELETE .
identityType	Required	String	The identity type that you are using to search the database. Use one of these values: ■ customer_id ■ login_id ■ subject_id Note: You cannot submit a GDPR delete request with other identity types (such as device_id or visitor_id),

Name	Required or Optional	Type	Description
			because they are not considered strong identifiers for an individual.
identityList	Required	List	The actual values of customer identities to delete. The value should match a real customer identity in the system. For example, if you specify a subject ID, data that is associated with the login ID and customer ID is also deleted for that customer. Follow these guidelines: ■ Values must be lowercase. ■ Values must match the type that is specified for identityType. The format for this value is similar to this example: <pre>["subject_id1", "subject_id2", "subject_id3"]</pre> In the output data, a blank row is returned if there is no data associated with that ID.

The response to the POST call contains the request ID for the delete job.

- 3 Use the request ID to view the request's status. Submit a GET call similar to this example:

```
GET https://<external_gateway_host>/marketingData/customerJobs/<delete_request_ID>
```

After the request is processed, all data and customer attributes for the specified identity types are deleted.

For more information about this example, see [“GDPR Delete of Identity Data” in SAS Customer Intelligence 360: Administration Guide](#).

Identity Records

Introduction to the REST API for Identities

To manage identity records that are stored in SAS Customer Intelligence 360 through a REST API, use endpoints in the Marketing Data API. With this REST API, you can perform these actions:

- submit a query based on an ID's value to retrieve the internal ID (data hub ID) that corresponds to that identity record
- return detailed information about an identity

The developer documentation for this API is available at [Marketing Data API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Data API, start with the correct URL domain.

- 1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingData/identityRecords
```

Example: Get the Details for a Known Identity

In this example, you send a series of REST calls to retrieve the detailed identity record for a known ID value (such as a customer ID).

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 85 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a GET call to retrieve the internal ID (the data hub ID) for a known ID type. In the GET call, define a query parameter based on the ID's value and type:
 - a Set the parameter to one of these values: `customerId`, `loginId`, or `visitorId`.
 - b Set the parameter equal to a known identity that matches the parameter's type. For example, if the parameter is `customerId`, the value should be a known customer ID.

For example, this call submits a query based on the customer ID value `qtmart_cust100043`:

```
GET https://<external gateway host>/marketingData/identityRecords?
customerId=qtmart_cust100043
```

In the response, make note of an identity's ID value. This is an example of a successful response:

```
{
  "id": "d6f0dd7a-c747-4a6b-b605-bd11195fe394",
  "version": 1,
  "links": [
    {
      "method": "GET",
      "rel": "self",
      "href": "https://<external gateway host>/marketingData/identityRecords/
d6f0dd7a-c747-4a6b-b605-bd11195fe394",
```

```

        "uri": "/identityRecords/d6f0dd7a-c747-4a6b-b605-bd11195fe394",
        "type": "application/vnd.sas.marketing.data.identityRecord"
    }
  ]
}

```

- 3 Send a GET call by using a data hub ID to retrieve other identity information that is associated with that customer:

GET `https://<external_gateway_host>/marketingData/identityRecords/<data_hub_ID>`

For example, here is a sample response that uses the data hub ID from the previous step (d6f0dd7a-c747-4a6b-b605-bd11195fe394):

```

{
  "id": "d6f0dd7a-c747-4a6b-b605-bd11195fe394",
  "version": 1,
  "identities": [
    {
      "id": "374942",
      "type": "subject_id",
      "source": "520af261-0108-4bc0-8dd9-623efe67aab8",
      "createTime": "2020-05-08 17:09:59.203",
      "attributes": {
        "cci.user_email": "Aayma.Baker@myEmail.com",
        "cci.recent_purchase_date": "2013-04-21 00:00:00.0",
        "cci.product_sub_start": "2018-11-20 00:00:00.0",
        "cci.country": "UnitedStates",
        "cci.zip": "02112",
        "cci.age": "21",
        "cci.product_sub_end": "2020-12-31 00:00:00.0",
        "cci.birthday": "1993-12-11 00:00:00.0",
        "cci.sign_up": "2018-03-09 00:00:00.0",
        "cci.city": "Boston",
        "cci.first_name": "Aayma",
        "cci.loyalty_sign_up": "2018-07-02 00:00:00.0",
        "cci.last_name": "Baker",
        "cci.trial_end": "2021-06-07 00:00:00.0",
        "cci.gender": "F",
        "cci.last_order": "2018-07-25 00:00:00.0",
        "cci.children": "N"
      }
    },
    {
      "id": "aayma.baker100043.lcj",
      "type": "login_id",
      "source": "520af261-0108-4bc0-8dd9-623efe67aab8",
      "createTime": "2020-05-08 17:09:59.202",
      "attributes": {
        "cci.user_email": "Aayma.Baker@myEmail.com",
        "cci.recent_purchase_date": "2013-04-21 00:00:00.0",
        "cci.product_sub_start": "2018-11-20 00:00:00.0",
        "cci.country": "UnitedStates",
        "cci.zip": "02112",
        "cci.age": "21",
        "cci.product_sub_end": "2020-12-31 00:00:00.0",
        "cci.birthday": "1993-12-11 00:00:00.0",
        "cci.sign_up": "2018-03-09 00:00:00.0",

```

```

        "cci.city": "Boston",
        "cci.first_name": "Aayma",
        "cci.loyalty_sign_up": "2018-07-02 00:00:00.0",
        "cci.last_name": "Baker",
        "cci.trial_end": "2021-06-07 00:00:00.0",
        "cci.gender": "F",
        "cci.last_order": "2018-07-25 00:00:00.0",
        "cci.children": "N"
    }
},
{
    "id": "8ad91916-cfb0-4354-b6a3-74618dee834e",
    "type": "visitor_id",
    "source": "import",
    "attributes": {
        "cci.user_email": "Aayma.Baker@myEmail.com",
        "cci.recent_purchase_date": "2013-04-21 00:00:00.0",
        "cci.product_sub_start": "2018-11-20 00:00:00.0",
        "cci.country": "UnitedStates",
        "cci.zip": "02112",
        "cci.age": "21",
        "cci.product_sub_end": "2020-12-31 00:00:00.0",
        "cci.birthday": "1993-12-11 00:00:00.0",
        "cci.sign_up": "2018-03-09 00:00:00.0",
        "cci.city": "Boston",
        "cci.first_name": "Aayma",
        "cci.loyalty_sign_up": "2018-07-02 00:00:00.0",
        "cci.last_name": "Baker",
        "cci.trial_end": "2021-06-07 00:00:00.0",
        "cci.gender": "F",
        "cci.last_order": "2018-07-25 00:00:00.0",
        "cci.children": "N"
    }
},
{
    "id": "qtmart_cust100043",
    "type": "customer_id",
    "source": "520af261-0108-4bc0-8dd9-623efe67aab8",
    "createTime": "2020-05-08 17:09:59.203",
    "attributes": {
        "cci.user_email": "Aayma.Baker@myEmail.com",
        "cci.recent_purchase_date": "2013-04-21 00:00:00.0",
        "cci.product_sub_start": "2018-11-20 00:00:00.0",
        "cci.country": "UnitedStates",
        "cci.zip": "02112",
        "cci.age": "21",
        "cci.product_sub_end": "2020-12-31 00:00:00.0",
        "cci.birthday": "1993-12-11 00:00:00.0",
        "cci.sign_up": "2018-03-09 00:00:00.0",
        "cci.city": "Boston",
        "cci.first_name": "Aayma",
        "cci.loyalty_sign_up": "2018-07-02 00:00:00.0",
        "cci.last_name": "Baker",
        "cci.trial_end": "2021-06-07 00:00:00.0",
        "cci.gender": "F",
        "cci.last_order": "2018-07-25 00:00:00.0",
    }
}

```

```

        "cci.children": "N"
      }
    },
    "links": [
      {
        "method": "GET",
        "rel": "self",
        "href": "https://<external_gateway_host>/marketingData/identityRecords/d6f0dd7a-c747-4a6b-b605-bd11195fe394",
        "uri": "/identityRecords/d6f0dd7a-c747-4a6b-b605-bd11195fe394",
        "type": "application/vnd.sas.marketing.data.identityRecord"
      }
    ]
  }
}

```

Marketing Execution API

About the Marketing Execution API

IMPORTANT The developer documentation for this API is available at [Marketing Execution API](#).

Introduction to the Marketing Execution REST API

The Marketing Execution REST API (marketingExecution) provides a way to execute bulk email tasks, bulk mobile tasks, direct marketing tasks, and on-premises segment maps. Executing an item via the REST API is the same action as clicking **Run Now** in a published item in the SAS Customer Intelligence 360 user interface.

After the task or on-premises segment map is executed, the REST API returns a URL that enables you to check the status of the execution (occurrence). This status gives you the ability to see whether the item executed successfully or failed, and the time stamps for when the task started and finished. Alternatively, you can go to the **Orchestration** ⇒ **History** ⇒ **Occurrence** page in the item to see a list of executions.

Endpoints in the Marketing Execution REST API

The root of the Marketing Execution API is /marketingExecution. Combine the root with one of these endpoints:

- /segmentMapJobs/ contains operations to create a job that executes a segment map and to retrieve details about a segment map job. For more information, see [“Segment Map Jobs”](#).
- /taskJobs contains operations to create a job that executes a bulk task or direct marketing task and to retrieve details about a task job. For more information, see [“Bulk Task Execution”](#).

- `/responseTrackingCodes` contains operations to manage Response Tracking Codes (RTCs). For more information, see [“Response Tracking Code \(RTC\)”](#).
- `/occurrences` contains operations to manage occurrences. For more information, see [“Occurrences”](#).

For examples of how to use these endpoints, see [“Examples for the Marketing Execution API” on page 209](#).

Prerequisites to Use the Marketing Execution API

If you do not already have a static JWT or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an access point so that you can generate a static JWT.
When you create the access point, make note of the **External gateway host** field on the Access Points page. This value is used as part of the URL for API calls that go through the external API gateway. Use this value when you see the variable `<external_gateway_host>` in the documentation.
- 2 (Optional) Create an API credential if you want to create a time-limited access token. This REST API does not require a temporary JWT, so you need to create an API credential only if you are completing the one-time setup to create an access token.
- 3 Make note of the value of the token that you need to use. The static JWT or access token must be specified in the Authorization header as a bearer token when you make calls to the REST API.

Examples for the Marketing Execution API

Segment Map Jobs

Introduction to the REST API for Segment Map Jobs

To execute jobs for segment maps through a REST API, use the `/segmentMapJobs/` endpoint of the Marketing Execution API. With this API, you can perform these actions:

- submit a job to execute a segment map
- access status about a submitted job

The developer documentation for this API is available at [Marketing Execution API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Execution API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.

- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable *<external gateway host>* in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingExecution/segmentMapJobs/<job ID>
```

Example: Submit a Job to Execute a Segment Map

Submit a job to execute an on-premises segment map. In this example, the name of the on-premises segment is *SampleSegmentMap*.

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 86 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a POST call to the `/segmentMapJobs/` endpoint:

```
POST https://<external gateway host>/marketingExecution/segmentMapJobs/
```

In the body of the request, specify the following information:

```
{
  "segmentMapId": "4ce189d7-b46f-4682-a38d-39bc4bdafdc4",
  "segmentMapName": "SampleSegmentMap",
  "folderPath": "ParentFolderName/ChildFolderName",
  "version": 1
}
```

This is an example of a successful response:

```
{
  "segmentMapId": "4ce189d7-b46f-4682-a38d-39bc4bdafdc4",
  "segmentMapName": "SampleSegmentMap",
  "version": 1,
  "segmentMapJobId": "c5ff01fb-0222-4ef7-8c4e-466fd0d416b6",
  "startedBy": "myuser",
  "startTimeStamp": "2021-06-15T10:30:15+0000",
  "endTimeStamp": "2021-06-15T10:31:41+0000",
  "executionState": "SUCCESS",
  "segmentMapJobType": "On demand",
  "objectType": "segment"
```

```
}
```

Bulk Task Execution

Building the Base URL for the API Calls

To successfully use the Marketing Execution API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable *<external gateway host>* in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingExecution/taskJobs/<taskId>
```

Example: Execute a Bulk Email Task

Use the `marketingExecution/taskJobs` endpoint to execute a task that is published.

Note: The `taskJobs` endpoint requires the `taskId` attribute or `taskName` attribute in the body of the request. If the task name is not unique, you must provide the `folderPath` attribute to specify which task name to execute.

To get a task ID, you can use the `tasks` endpoint of the [Marketing Design on page 217](#) REST API.

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 87 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 Create the POST call to the Marketing Execution API to execute the task. In this example, the body of the call contains the `taskName` attribute and the value `bulk-holiday-email`. Here is an example of the POST call:

```
POST https://<external gateway host>/marketingExecution/taskJobs
```

Here is an example of the body that uses the `taskId` value from the previous step. In addition, this body sets the `overrideSchedule` attribute to `true`. This value is required when you want to run a scheduled task or scheduled segment from the REST API, because the REST call will override any schedule that is specified in the **Orchestration** ⇒ **Schedule** page of the task.

```
{
  "taskId": "bulk-holiday-email",
  "overrideSchedule": true
}
```

A successful response contains the `taskJobId` attribute. You can use this value to check on the progress of the execution. Here is an example of a successful response:

```
{
  "version": 1,
  "taskJobId": "decbcf0b-7b20",
  "startedBy": "user",
  "startedTimeStamp": "2021-07-12T19:47:06+0000",
  "executionState": "In progress",
  "taskJobType": "On demand",
  "taskId": "bdc-dff8fae2",
  "objectType": "task",
  "links": [...]
}
```

- 3 (Optional) Check the status of the task execution. Send a GET call to the [marketingDesign/taskJobs](#) on page 212 endpoint using the `taskJobId` value that is provided in the response from the previous step.

GET `https://<external_gateway_host>/marketingDesign/taskJobs/<taskJobId>`

For example, to get the execution status of the bulk-holiday-email task, use a call similar to this one:

GET `https://<external_gateway_host>/marketingDesign/taskJobs/decbcf0b-7b20`

TIP You can see this occurrence by signing in to the product and navigating to the **Orchestration** ⇒ **History** tab of the task and clicking **Occurrences**.

Occurrences

Introduction to the REST API for Occurrences

An *occurrence* is a single execution of an item and is generated either on a recurring schedule or from a user who manually runs the item. You can see a list of occurrences for an item on the **Orchestration** ⇒ **History** ⇒ **Occurrences** tab of that item. To retrieve occurrence information through an API, use the Marketing Execution API endpoint for occurrences (`/occurrences`).

Currently, only these items generate occurrences:

- bulk tasks (direct marketing, email, and mobile push)
- on-premises segment maps

The developer documentation for this API is available at [Marketing Execution API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Execution API, start with the correct URL domain.

- 1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external_gateway_host>/marketingExecution/occurrences
```

Example: Get the Execution History of an Item

To retrieve occurrence details for a task or segment map that is based on an occurrence ID:

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 88 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 If you do not have the occurrence ID, you can get a list of occurrences and their IDs by using this API call:

```
GET https://<external_gateway_host>/marketingExecution/occurrences
```

TIP You can filter the number of occurrences in the response by using one of the available parameters. For example, you can return occurrences for tasks by using this API call:

```
GET https://<external_gateway_host>/marketingExecution/occurrences?type=task
```

See the developer documentation for a complete list of parameters and valid values.

- 3 Send a GET call to retrieve the occurrence information for the item by using the occurrence ID.

```
GET https://<external_gateway_host>/marketingExecution/occurrences/<occurrence ID>
```

For example, to retrieve an occurrence with an ID of `976f153850b9`, use this API call:

GET `https://<external_gateway_host>/marketingExecution/occurrences/976f153850b9`

This is an example of a successful response:

```
{
  "occurrenceNumber": 3,
  "startTime": "1970-01-01T00:00:00+0000",
  "occurrenceId": "976f153850b9",
  "endTime": "1970-01-01T00:00:00+0000",
  "properties": {},
  "objectId": "aut_20200220094400556794",
  "objectType": "task"
}
```

Response Tracking Code (RTC)

Introduction to the REST API for RTCs

To retrieve metadata about [response tracking codes \(RTCs\)](#) through a REST API, use the Marketing Execution API endpoint for response tracking codes (`/responseTrackingCodes`). An *RTC* is an automatically generated code that represents the unique combination of items that an individual is shown. An RTC provides a link between the content that a user was presented and the content that the user responded to. RTCs can also help you understand which content was shown to an individual.

The developer documentation for this API is available at [Marketing Execution API](#).

Data Tables That Contain RTC Information

RTC information is stored in contact and response data tables that are available for download. You can use the RTC tables as a key for analyzing data in other tables that contain information about contacts, responses, and metadata for a specific RTC.

- Metadata about an RTC is contained in the UDM table [MD_RTC](#) as a part of schemaVersion 6.
- Occurrence metadata is contained in the UDM table [MD_OCCURRENCE](#) as a part of schemaVersion 6.

These tables contain RTC information. They include the RTC value in the `RESPONSE_TRACKING_CD` column:

- [CONTACT_HISTORY](#)
- [RESPONSE_HISTORY](#)

These tables contain RTC information and are available if CDM is enabled. They include the RTC value in the `RTC_ID` column:

- [CDM_RTC_DETAIL](#)
- [CDM_RTC_X_CONTENT](#)
- [CDM_CONTACT_HISTORY](#)
- [CDM_RESPONSE_HISTORY](#)

Building the Base URL for the API Calls

To successfully use the Marketing Execution API, start with the correct URL domain.

- 1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingExecution/responseTrackingCodes
```

Example: Get a List of Response Tracking Codes

To retrieve information about a response tracking code by its ID, follow these steps. This retrieval is useful in situations in which you have an RTC ID from a streamed event and need to know more details about the RTC, such as the content that is associated with that RTC. For example, if you have the RTC ID from a contact or response event, you can use that ID to find out which messages or creatives are associated with that RTC.

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 89 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGci0.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a GET call to retrieve details about an RTC ID.

```
GET https://<external gateway host>/marketingExecution/responseTrackingCodes/
<response tracking code ID>
```

For example, to retrieve details about an RTC with an ID of `d90a2c296be0`, use this API call:

```
GET https://<external gateway host>/marketingExecution/responseTrackingCodes/
d90a2c296be0
```

This is an example of a successful response:

```
{
  "rtcId": "d90a2c296be0",
  "creatives": [
    {
      "creativeVersionId": "8q5GQYNu1xILSa6tbnmpXq_",
      "version": 1,
      "creativeId": "0db4fc6c-a09b195cc-44b5dd87"
    }
  ],
}
```

```

    "messages": [
      {
        "messageVersionId": "2nXsundaymotJ1bARq3TMb81EJH",
        "messageId": "40109446-3419-42f2-29b495e703f6",
        "version": 2
      }
    ],
    "rtcTimestamp": "2021-02-15T15:22:09+0000",
    "version": 2,
    "taskId": "3aa8dcd7-44d9-4144-d38c078bf9",
    "taskVersionId": "orqX19bpQkASKQ8hILJ4u0"
  }
}

```

Example: Get a List of Response Tracking Codes for a Specific Task

To retrieve a list of response tracking codes for a specific task based on the task ID:

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 90 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v

- 2 If you do not have the ID for the task that you want to retrieve the RTC for, you can get a list of tasks and their IDs by using the Marketing Execution API endpoint for tasks.

GET https://<external gateway host>/marketingExecution/tasks

TIP You can filter the number of tasks in the response by using one of the available parameters. For example, you can return tasks that match a channel such as web or mobile. For more information, see “Tasks” on page 219.

- 3 Send a GET call to retrieve the RTC information for the task by using the task ID.

GET https://<external gateway host>/marketingExecution/responseTrackingCodes?
taskId=<task ID>

For example, to retrieve a task with an ID of *3aa844d9-414-d38c0*, use this API call:

GET https://<external gateway host>/marketingExecution/responseTrackingCodes?
taskId=3aa844d9-414-d38c0

This is an example of a successful response:

```

{
  "responseTrackingCodes": [

```

```

{
  "rtcId": "00674152-6340-b08d-d90a2c",
  "creatives": [
    {
      "creativeVersionId": "8q5GQY1xILSa6tbnmpXq_",
      "version": 1,
      "creativeId": "0db4fc6c-a09b-95cc-44a23c7"
    }
  ],
  "messages": [
    {
      "messageVersionId": "2nXXnaH4FotJ1bARq3TMb81EJH",
      "messageId": "401446-3419-29b403f6",
      "version": 2
    }
  ],
  "rtcTimestamp": "2021-02-15T15:22:09+0000",
  "version": 2,
  "taskId": "3aa844d9-414-d38c0",
  "taskVersionId": "orqXpQl_JkA9fwAhILJ4uO"
},
"count": 1,
"countScanned": 2735

```

Marketing Design API

Marketing Design API

Introduction to the Marketing Design API

The Marketing Design API provides a way to access and view information about tasks and on-premises segment maps in SAS Customer Intelligence 360. For example, you can use this API to access the following information:

- a task's channel such as web, mobile, email, direct, external, and so on
- the details for a task, such as name, ID, priority, publish date, last modified date, and so on
- the status of a task, such as Designing, Active, Completed, and so on
- the task's usage information about dependent items such as spots, events, creatives, messages, and so on
- the details for a segment map, such as its status, business context, and published date

TIP When you access information about tasks, you can use optional parameters to filter and sort the list of tasks. For example, you can sort responses based on criteria such as task type or date modified.

The developer documentation for this API is available at [Marketing Design API](#).

Endpoints in the Marketing Design API

The root of the Marketing Design API is `/marketingDesign`. Combine the root with one of these endpoints:

- `/tasks/` contains operations to manage tasks. For more information, see [“Tasks”](#).
- `/segmentMaps/` contains operations to retrieve summary information for segment maps and detailed data for a specific segment map. For more information, see [“Segment Maps” on page 221](#).
- (Deprecated) `/directMarketingTasks/` contains operations to create jobs for direct marketing tasks and to retrieve their status. This endpoint is deprecated, and the functionality is in the `/taskJobs` endpoint of the Marketing Execution API. For more information, see [“Bulk Task Execution”](#).

To view information about direct marketing tasks, use the `/tasks/` endpoint of the Marketing Design API.

- (Deprecated) `/occurrences` contains operations to manage occurrences. This endpoint is deprecated, and the functionality is now part of the Marketing Execution API. For more information, see [“Occurrences”](#).
- (Deprecated) `/responseTrackingCodes` contains operations to manage Response Tracking Codes (RTCs). This endpoint is deprecated, and the functionality is now part of the Marketing Execution API. For more information, see [“Response Tracking Code \(RTC\)”](#).

For examples of how to use these endpoints, see [“Examples for the Marketing Design API” on page 219](#).

Prerequisites to Use the Marketing Design API

If you do not already have a static JWT or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an access point so that you can generate a static JWT.
When you create the access point, make note of the **External gateway host** field on the Access Points page. This value is used as part of the URL for API calls that go through the external API gateway. Use this value when you see the variable `<external_gateway_host>` in the documentation.
- 2 (Optional) Create an API credential if you want to create a time-limited access token. This REST API does not require a temporary JWT, so you need to create an API credential only if you are completing the one-time setup to create an access token.
- 3 Make note of the value of the token that you need to use. The static JWT or access token must be specified in the Authorization header as a bearer token when you make calls to the REST API.

Examples for the Marketing Design API

Tasks

Introduction to the REST API for Tasks

To access information about tasks through a REST API, use the `/tasks` endpoint of the Marketing Design API. These are the types of task details that you can access with this API:

- the channel of a task, such as direct, web, mobile, email, and external
- details about a task, such as name, ID, priority, publish date, and last modified date
- the status of a task, such as Designing, Active, and Completed
- information about the items that are used in a task, as well as which items the task depends on

The developer documentation for this API is available at [Marketing Design API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Design API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external_gateway_host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

`GET https://<external_gateway_host>/marketingDesign/tasks/?channel=direct`

Example: Get a List of Mobile Tasks

To retrieve a list of mobile tasks:

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 91 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOi0.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the list of mobile tasks:

GET https://<external gateway host>/marketingDesign/tasks?channel=mobile

This is an example of a successful response:

```
"links": [...],
  "name": "tasksSummaryStandard",
  "accept": "application/vnd.sas.marketing.design.tasks.summary",
  "count": 3,
  "items": [
    {
      "version": 1,
      "taskId": "612cb022-50a3-497d-bc72-1cfdd3d58110",
      "name": "rnMobileTask",
      "lastModifiedTimeStamp": "Fri May 31 18:51:24 UTC 2019",
      "lastPublishedTimeStamp": "Fri May 31 18:51:23 UTC 2019",
      "state": "active",
      "channel": "mobile",
      "taskType": "single",
      "numSpots": 1,
      "deliveryType": null,
      "priority": 3
    },
    {
      "version": 1,
      "taskId": "f9ed0e38-d45f-44a1-9f9e-c1fc48f3dc02",
      "name": "lks_iris_trigpn",
      "lastModifiedTimeStamp": "Fri Jan 24 20:06:58 UTC 2020",
      "lastPublishedTimeStamp": null,
      "state": "designing",
      "channel": "mobile",
      "taskType": "single",
      "numSpots": 1,
      "deliveryType": "triggered",
      "priority": 3
    },
    {
      "version": 1,
      "taskId": "030306ad-401a-45d1-b293-f68d389135b7",
      "name": "MPVEvent_PVETask",
      "lastModifiedTimeStamp": "Tue Jun 16 21:01:09 UTC 2020",
      "lastPublishedTimeStamp": "Tue Jun 16 21:01:09 UTC 2020",
      "state": "active",
      "channel": "mobile",
      "taskType": "single",
      "numSpots": 1,
      "deliveryType": null,
```

```

        "priority": 3
      },
    ],
    "version": 2
  }

```

Segment Maps

Introduction to the REST API for Segment Maps

To access information about segment maps through a REST API, use the `/segmentMaps` endpoint of the Marketing Design API. These are the types of segment map details that you can access with this API:

- summary information for a collection of segment maps
- details about a segment map, such as name, ID, status, publish date, and last modified date

The developer documentation for this API is available at [Marketing Design API](#).

Building the Base URL for the API Calls

To successfully use the Marketing Design API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external gateway host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external gateway host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external gateway host>/marketingDesign/segmentMaps/<segment map ID>
```

Example: Get Details for a Segment Map

To retrieve details for a segment map:

- 1 To authenticate with the REST API, specify your authorization token as the bearer token in all REST API calls. This token might be a static JWT or an access token.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 92 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOi0.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the details for a segment map:

GET https://<external gateway host>/marketingDesign/segmentMaps/91e02210-c418-4f88-a50d-327b451f6571

This is an example of a successful response:

```
{
  "id": "91e02210-c418-4f88-a50d-327b451f6571",
  "name": "simpleSegMap3",
  "code": "MAP_244",
  "folderName": "/",
  "status": "active",
  "businessContextId": "fc2a96da-4c48-42af-947e-4c020df87dc2",
  "createdByUserName": "Joe.Smith@sas.com",
  "modifiedByUserName": "Joe.Smith@sas.com@opdmstr",
  "createdDate": "2020-12-18T19:50:41Z",
  "modifiedDate": "2021-10-18T19:14:27.738Z",
  "publishedDate": "2021-10-18T19:14:29.028Z",
  "customAttributes": [
    {
      "name": "month",
      "value": "February",
      "type": "TEXT"
    }
  ],
  "segments": [
    {
      "id": "cb7eeb20-8b55-467a-af43-c9ffefef334f2",
      "name": "Age=58",
      "code": "SEG_536",
      "customAttributes": [
        {
          "name": "campaign",
          "value": "monthly",
          "type": "TEXT"
        }
      ],
      "lastExecutionCount": 6181
    }
  ],
  "versions": [
    {
      "id": "WKCma4.UAFLU8nh.6N_f1gD8WWHa098G",
      "validFrom": "2021-10-18T19:14:30Z",
      "validTo": null,
      "latest": true
    }
  ]
}
```

```

],
"lastExecutionDate": "2026-04-13T15:24:08.447Z",
"lastExecutionCount": 6181,
"lastExecutionStatus": "Success"
}

```

Marketing Gateway API

About the Marketing Gateway API

IMPORTANT The developer documentation for this API is available at [Marketing Gateway API](#).

Introduction to the Marketing Gateway API

The Marketing Gateway API enables you to retrieve event data, examine your access configuration, and stream events. For example, you can use this API to perform these actions:

- review configuration settings for your access point.
- download .zip files for on-premises agents.
- download data tables for events, customer identities, reports, and published items (metadata).
- inject external events. You can batch process events by uploading a CSV file, or you can inject single events in the body of the REST call.

Endpoints in the Marketing Gateway API

The root of the Marketing Gateway API is `/marketingGateway`. Combine the root with one of these endpoints:

- `/bulkEventsFileLocation` contains operations to batch upload events from a CSV file.
- `/configuration` contains operations to retrieve the settings for the access point and tenant-level properties for your environment.
- `/discoverService/dataDownload/eventData/` contains operations and endpoints that enable you to download data tables from SAS Customer Intelligence 360.
- `/events` contains operations to inject single events into SAS Customer Intelligence 360.

For examples of how to use these endpoints, see [“Examples for the Marketing Gateway API”](#) on page 224.

Prerequisites to Use the Marketing Gateway API

If you do not already have a static JWT or an access token, you must [configure the authentication method](#). Complete these steps as part of the authentication setup:

- 1 Create an access point so that you can generate a static JWT.
When you create the access point, make note of the **External gateway host** field on the Access Points page. This value is used as part of the URL for API calls that go through the external API gateway. Use this value when you see the variable `<external_gateway_host>` in the documentation.
- 2 (Optional) Create an API credential if you want to create a time-limited access token. This REST API does not require a temporary JWT, so you need to create an API credential only if you are completing the one-time setup to create an access token.
- 3 Make note of the value of the token that you need to use. The static JWT or access token must be specified in the Authorization header as a bearer token when you make calls to the REST API.

Examples for the Marketing Gateway API

- To batch load events from a CSV file, see [“Using the REST API to Batch Load External Events” in SAS Customer Intelligence 360: Administration Guide](#).
- To download data tables from SAS Customer Intelligence 360, see one of these topics:
 - [“Downloading Data Tables with the REST API” in SAS Customer Intelligence 360: Administration Guide](#)
 - [Data Download](#) in the developer documentation
- To inject single events into SAS Customer Intelligence 360, see [“Injecting an External Event into SAS Customer Intelligence 360” in SAS Customer Intelligence 360: Administration Guide](#).

Plan API

About the Plan API

IMPORTANT The developer documentation for this REST API is available at [Plan API](#).

Introduction to the Plan API

The Plan API enables you to create, read, update, and delete a variety of resources that are related to SAS 360 Plan. Use the Plan API endpoint (`/marketingPlan`) to perform these actions:

- track all marketing efforts across various channels instead of using offline spreadsheets and documents
- capture the objectives and goals of marketing efforts throughout the year
- ensure alignment of overall marketing efforts to the organization's strategic plan and objectives
- provide visibility of marketing plans, programs, activities, and tasks to stakeholders within the organization

Endpoints in the Plan API

The root of the Marketing Plan API is `/marketingPlan`. Combine the root with one of these endpoints:

- `/hierarchycomponents/schemas/<hierarchy definition ID>/hierarchyDefinitionLevels` contains operations to manage hierarchy levels for a hierarchy definition. For more information, see [Hierarchy Definition Levels](#) in the developer documentation.
- `/hierarchyDefinitions` and `/hierarchycomponents/schemas/{hierarchyDefinitionId}` contain operations to manage hierarchy definitions and view details about them. For more information, see [Hierarchy Definitions](#) in the developer documentation.
- `/planningItems` contains operations to create, update, delete, and view plan items. For more information, see [“Managing Planning Items”](#).
- `/settings` contains operations to manage settings for SAS 360 Plan. For more information, see [Settings](#) in the developer documentation.

Prerequisites to Use the Plan API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see [“Setting Up Authorization” in SAS Customer Intelligence 360: Administration Guide](#).

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.

When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable `<external gateway host>` in the documentation.
 - b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.

- 2 Generate the temporary JWT with your static JWT or the `client_jwt` value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the Plan API

Managing Planning Items

About the REST API Endpoint for Planning Items

To use the REST API to manage planning items, use the endpoints that are part of the Plan API (`/marketingPlan`). With this REST API, you can perform these actions:

- track all marketing efforts across various channels instead of using offline spreadsheets and documents
- capture the objectives and goals of marketing efforts throughout the year
- ensure alignment of overall marketing efforts to the organization's strategic plan and objectives
- provide visibility of marketing plans, programs, activities, and tasks to stakeholders within the organization

Building the Base URL for the API Calls

To successfully use the Plan API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:
GET `https://<external_gateway_host>/marketingPlan/hierarchyDefinitions`

Example: Get the Details of Planning Item

In this example, you can get the details of specific planning items. First, you send a REST call to retrieve the list of planning items. Then, you use an ID from a planning item to retrieve the planning item's information.

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 93 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the list of planning items:

GET https://<external gateway host>/marketingPlan/planningItems

In the response, make note of the planning item's ID. The response is similar to this output:

```
{
  "links": [ ... ],
  "name": "PlannerInstances",
  "accept": "application/vnd.sas.marketing.planning.item.summary",
  "start": 0,
  "count": 2,
  "items": [
    {
      "version": 2,
      "id": "5dbf163e-3135-4222-9f1b-d8e99f04bae2",
      "name": "My_plan_1",
      "modifiedTimeStamp": "2021-04-01T05:32:44.562Z",
      "modifiedBy": "sasUser",
      "modifiedByName": "sasUser",
      "hierarchyDefinitionId": "a9744a67-9c14-4c44-acff-2e6c391d4cc2",
      "hierarchyDefinitionLevelId": "b23eae5-cd23-4bcd-860e-422a6628e162",
      "plannedStartDate": "2020-02-24T19:20:45.000Z",
      "plannedEndDate": "2020-02-28T19:20:45.000Z",
      "hierarchyLevel": "Plan",
      "type": "planningItems",
      "statusLocalized": "Planning",
      "currencyCode": "USD",
      "typeLabel": "Plan",
      "state": "planning",
      "createdBy": "sasUser",
      "createdByName": "sasUser",
      "number": "PLN-881",
      "oldObject": false,
      "publicAccessDenied": false,
      "links": [...]
    },
    ...
  ],
  "limit": 50,
  "version": 2
}
```

3 Using the planning item's ID from the previous step, retrieve the planning item details:

GET https://<external_gateway_host>/marketingPlan/planningItems/
5dbf163e-3135-4222-9f1b-d8e99f04bae2

This is an example of a successful response:

```
{
  "version": 2,
  "id": "5dbf163e-3135-4222-9f1b-d8e99f04bae2",
  "number": "Plan-1003",
  "name": "My_plan_1",
  "modifiedTimeStamp": "2021-04-01T05:32:44.562Z",
  "modifiedBy": "sasUser",
  "modifiedByName": "sasUser",
  "hierarchyDefinitionId": "a9744a67-9c14-4c44-acff-2e6c391d4cc2",
  "hierarchyDefinitionLevelId": "b23eaae5-cd23-4bcd-860e-422a6628e162",
  "plannedStartDate": "2020-02-24T19:20:45.000Z",
  "plannedEndDate": "2020-02-28T19:20:45.000Z",
  "hierarchyLevel": "Plan",
  "type": "planningItems",
  "statusLocalized": "Planning",
  "childCount": 0,
  "typeLabel": "Plan",
  "state": "planning",
  "customAttributes": {
    "groups": [
      {
        "label": "Master-Marketing & Product Information",
        "visible": true,
        "obsolete": false,
        "fields": [
          {
            "label": "Product Category",
            "value": "Traditional IRA",
            "attributeCode": "Prod_Category",
            "type": "Text",
            "visible": true,
            "obsolete": false,
            "adHoc": false,
            "validator": {
              "required": true
            },
            "displayOrder": 0,
            "dynamic": false
          },
          {
            "description": "max chars=100",
            "label": "Additional Product Information",
            "attributeCode": "Additional_Prod_Info",
            "type": "Text",
            "visible": true,
            "obsolete": false,
            "adHoc": false,
            "constraints": {
              "maxLength": 100,
              "multiLine": true
            },
            "displayOrder": 0,

```

```

        "dynamic": false
      }
    ],
    "groupId": "f30d3ac3-738b-4460-93f1-489f9f8904fa"
  }
]
},
"tags": [
  {
    "id": "ada02707-bc4f-447e-9f8f-d51f953dbc9a",
    "name": "smu-pgm",
    "createdBy": "sasUser",
    "createdDate": "2018-08-13T08:41:33.155Z",
    "lastModifiedDate": "2018-08-13T08:41:33.155Z",
    "lastModifiedBy": "sasUser",
    "modifiedStatusCode": "CREATED",
    "count": 0,
    "links": [
      {
        "method": "GET",
        "rel": "self",
        "href": "/tags/ada02707-bc4f-447e-9f8f-d51f953dbc9a",
        "uri": "/tags/ada02707-bc4f-447e-9f8f-d51f953dbc9a",
        "type": "application/vnd.sas.marketing.tag"
      }
    ]
  }
],
"links": [
  {
    "method": "GET",
    "rel": "self",
    "href": "/planningItems/5dbf163e-3135-4222-9f1b-d8e99f04bae2",
    "uri": "/planningItems/5dbf163e-3135-4222-9f1b-d8e99f04bae2",
    "type": "application/vnd.sas.marketing.planning.item"
  },
  {
    "method": "GET",
    "rel": "children",
    "href": "/planningItems/5dbf163e-3135-4222-9f1b-d8e99f04bae2/children?start=0&limit=50",
    "uri": "/planningItems/5dbf163e-3135-4222-9f1b-d8e99f04bae2/children?start=0&limit=50",
    "type": "application/vnd.sas.collection",
    "itemType": "application/vnd.sas.marketing.planning.item.summary"
  },
  {
    "method": "DELETE",
    "rel": "delete",
    "href": "/planningItems/5dbf163e-3135-4222-9f1b-d8e99f04bae2",
    "uri": "/planningItems/5dbf163e-3135-4222-9f1b-d8e99f04bae2"
  }
]
}

```

System for Cross-Domain Identity Management (SCIM) API

About the System for Cross-Domain Identity Management (SCIM) API

IMPORTANT The developer documentation for this API is available at [SCIM API](#).

Introduction to the SCIM API

The System for Cross-domain Identity Management (SCIM) API for SAS Customer Intelligence 360 is a centralized, web-based repository that enables an organization to manage identity resources such as users, groups, and roles. The SCIM REST API is based on the SCIM industry standard. Use this API to complete actions related to users, groups, and roles:

- get information about users
- create or delete a user, change the roles assigned to the users, or change group membership for a user
- get information about groups
- create, modify, and delete groups
- get information about roles, including a list of roles
- manage group membership and the roles that are assigned to groups

Note: The SCIM REST API can manage accounts only in the SAS Customer Intelligence 360 application. The REST API does not create or manage the single sign-on (SSO) accounts for your authentication provider.

Endpoints in the SCIM API

The root of the SCIM API is `/scim/v2`. Combine the root with one of these endpoints:

- `/Groups` contains operations to manage groups. For more information, see [Groups](#) in the developer documentation.
- `/Roles` contains operations to manage user roles. For more information, see [Roles](#) in the developer documentation.
- `/ServiceProviderConfig` contains operations for configuration and schemas. For more information, see [Configuration and Schemas](#) in the developer documentation.

- `/Users` contains operations to manage users. For more information, see [Users](#) in the developer documentation.

Prerequisites to Use the SCIM API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see [“Setting Up Authorization” in SAS Customer Intelligence 360: Administration Guide](#).

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.
When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable `<external_gateway_host>` in the documentation.
 - b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.
- 2 Generate the temporary JWT with your static JWT or the `client_jwt` value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the SCIM API

List Users

For more examples and details about the SCIM API, see the developer documentation at [SCIM API](#).

Building the Base URL for the API Calls

To successfully use the SCIM API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

`https://<external_gateway_host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

GET `https://<external_gateway_host>/scim/v2/Users`

Example: List Users

To retrieve a list of users, follow these steps:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 94 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer <code>eyJ0eXAiOiGciO.DAxMGFhODEJ9.BRv_nD6v</code>

- 2 Send a GET call to retrieve the list of users:

GET `https://<external_gateway_host>/scim/v2/Users`

This is an example of a successful response:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "startIndex": 1,
  "itemsPerPage": 50,
  "totalResults": 2,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"
      ],
      "id": "ab3b9de3-61ff-429a-be8f-c2270ed43726",
      "userName": "user1@email.com",
      "admin": false,
      "meta": {
        "resourceType": "User",
        "created": "2020-03-02T14:29:49.000Z",
        "lastModified": "2020-05-07T19:01:32.000Z",
        "location": "https://<external_gateway_host>/scim/v2/Users/ab3b9de3-be8f-c2270ed43726"
      }
    },
    {
      "schemas": [
```

```

        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"
    ],
    "id": "2b487ede-3b18-465a-9ccb-e2cdf1af92e8",
    "userName": "user2@email.com",
    "admin": false,
    "meta": {
        "resourceType": "User",
        "created": "2020-11-05T18:27:24.000Z",
        "lastModified": "2020-11-05T18:27:24.000Z",
        "location": "https://<external gateway host>/scim/v2/Users/
2b487ede-9ccb-e2cdf1af92e8"
    }
},
}

```

Find Roles for Users

For more examples and details about the SCIM API, see the developer documentation at [SCIM API](#).

Building the Base URL for the API Calls

To successfully use the SCIM API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable *<external gateway host>* in the documentation.

An example might use the following URL for an endpoint:

`https://<external gateway host>/<endpoint>`

- 2 Use this base URL with the API paths to make calls. For example:

`GET https://<external gateway host>/scim/v2/Users`

Example: Find the Roles Associated with a User

To retrieve the roles that are associated with a specific user, you can send a GET call with the *filter* parameter. In this example, the *filter* parameter uses these criteria:

- `userName eq "user1@email.com"` looks for the user whose user name matches `user1@email.com`.
- `attributes=roles` returns the roles that are associated with the user.

Follow these steps to send a GET call that returns the roles that are associated with a user:

- 1 Make sure that you generated a temporary access token. Specify the temporary access token as the value for the Authorization header in the REST API call. Use the term "Bearer" as a prefix and add a space before the value of the token.
- 2 Send a GET call to retrieve the roles associated with a user:

GET https://<external_gateway_host>/scim/v2/Users?filter=username eq
"user1@email.com"&attributes=roles

This is an example of a successful response:

```
{
  "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ],
  "startIndex": 1,
  "itemsPerPage": 1,
  "totalResults": 1,
  "Resources": [
    {
      "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:User",
        "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"
      ],
      "id": "3dd7a567-375e-4999-b741-5ae827af47ed",
      "userName": "user1@email.com",
      "admin": false,
      "roles": [
        {
          "value": "38b6c9e0-8c56-4473-b806-e89e422b18be",
          "roleName": "Discover Admin",
          "$ref": "https://<external_gateway_host>/scim/v2/Roles/38b6c9e0-8c56-e89e422b18be"
        },
        {
          "value": "aeb50ff0-15bb-4ec5-b3e8-7987763d5040",
          "roleName": "Engage Viewer",
          "$ref": "https://<external_gateway_host>/scim/v2/Roles/aeb50ff0-15bb-7987763d5040"
        },
        {
          "value": "938565f1-7cdb-40ab-aa1-6c76511d0a4a",
          "roleName": "Insights Viewer",
          "$ref": "https://<external_gateway_host>/scim/v2/Roles/938565f1-7cdb-6c76511d0a4a"
        }
      ],
      "meta": {
        "resourceType": "User",
        "created": "2019-04-10T13:12:00.000Z",
        "lastModified": "2020-05-07T19:01:38.000Z",
        "location": "https://<external_gateway_host>/scim/v2/Users/3dd7a567-375e-5ae827af47ed"
      }
    }
  ]
}
```

Workflow API

About the Workflow API

IMPORTANT The developer documentation for this REST API is available at [Workflow API](#).

Introduction to the Workflow API

Workflows enable an organization to automate processes and save time. You can use automated workflows to manage assignments and provide real-time visibility to project timelines. Workflows contain workflow tasks, which are assigned to contributors (users or groups) that complete the tasks. Use the Workflow API endpoint (`marketingWorkflow`) to perform these actions:

- create and launch a workflow
- search for a workflow template
- search for a workflow
- add workflow tasks to a workflow
- assign workflow tasks to contributors (users and groups) and initiate the workflow tasks
- cancel, complete, approve, or reject workflow tasks
- attach and upload files as supporting resources in a workflow task
- download attachments

Endpoints in the Workflow API

The root of the Workflow API is `/marketingWorkflow`. Combine the root with one of these endpoints:

- `/definitions` contains operations that enable you to retrieve information about existing workflow definitions. For more information, see [“Workflow Definitions”](#).
- `/processes` contains operations that enable you to create workflow processes and manage existing processes. For more information, see [Processes](#) in the developer documentation.

In addition, this endpoint contains additional endpoints to manage items for a workflow process:

- `/processes/<process ID>/tasks` contains operations that enable you to manage workflow tasks. For more information, see [Get a collection of workflow tasks](#) in the developer documentation.
- `/processes/<process ID>/tasks/<task ID>/attachments` contains operations that create attachments and manage existing attachments for a task. For more information, see [Attachments](#) in the developer documentation.

- `/taskAssignees/<task assignee ID>` contains operations to retrieve information about a user that is assigned to workflow items. For more information, see [Assignments](#) in the developer documentation.
- `/tasks` contains operations that enable you to update or retrieve information for existing tasks. For more information, see [Tasks](#) in the developer documentation.

For examples of how to use these endpoints, see “[Examples for the Workflow API](#)” on page 236.

Prerequisites to Use the Workflow API

To authenticate with this REST API, you must use a temporary JSON Web Token (JWT) that you generate from the `/token` endpoint. You can generate the temporary JWT with a static JWT or from an existing access token. For more information, see “[Setting Up Authorization](#)” in *SAS Customer Intelligence 360: Administration Guide*.

You might need to complete these steps as part of the authentication setup.

- 1 If you are not using an access token, complete these steps:
 - a Create an access point if one does not exist. You must have an access point to generate a static JWT, which is used to generate the temporary JWT.

When you create the access point, make note of the value in the **External gateway host** field on the Access Points page. Use this value when you see the variable `<external gateway host>` in the documentation.
 - b Create an API credential if one does not exist. API credentials are required to generate a temporary JWT.
- 2 Generate the temporary JWT with your static JWT or the `client_jwt` value from your access token.
- 3 When you make calls to the REST API, use the temporary JWT as the bearer token in the Authorization header.

Examples for the Workflow API

Workflow Definitions

Introduction to the REST API for Workflow Definitions

To manage workflow definitions with the REST API, you can use the `/definitions` endpoint in the Workflow API (`/marketingWorkflow`). With this REST API, you can perform these actions:

- retrieve the list of workflow definitions
- retrieve information about a specific workflow definition

The developer documentation for this REST API is available at [Workflow API](#).

Building the Base URL for the API Calls

To successfully use the Workflow API, start with the correct URL domain.

1 Find the host name for the REST API.

- a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
- b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external_gateway_host>/marketingWorkflow/definitions
```

Example: Get the Details for a Workflow Definition

In this example, you can get the details for a workflow definition. First, you send a REST call to retrieve the list of workflow definitions. Then, you use an ID from a definition to retrieve that definition's details.

1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 95 Using a Token as the Authorization Header

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGciO.DAxMGFhODEJ9.BRv_nD6v

2 Send a GET call to retrieve the list of workflow definitions:

```
GET https://<external_gateway_host>/marketingWorkflow/definitions
```

In the response, make note of the workflow definition's ID. The response is similar to this output:

```
{
  "links": [ ... ],
  "name": "WorkflowDefinitions",
  "accept": "application/json application/
vnd.sas.marketing.workflow.definition.summary+json application/vnd.sas.collection
+json",
  "start": 0,
  "count": 10,
  "items": [
    {
      "version": 1,
```

```

    "id": "4797816a-45c4-4605-8088-439a138fee29",
    "name": "Workflow Template C995443",
    "description": "",
    "lastModifiedTimeStamp": "2021-04-01T04:54:53Z",
    "lastModifiedBy": "sasUser",
    "lastModifiedByName": "sasUser",
    "associatedObjectTypes": "Planner.PlanningObject",
    "workflowType": "GENERIC",
    "disabled": false,
    "filesAllowedToBeCataloged": false,
    "state": "DRAFT",
    "versionNumber": 1,
    "links": [
      {
        "method": "GET",
        "rel": "self",
        "href": "https://<external gateway host>/marketingWorkflow/
definitions/4797816a-45c4-4605-8088-439a138fee29",
        "uri": "/definitions/4797816a-45c4-4605-8088-439a138fee29",
        "type": "application/vnd.sas.marketing.workflow.definition"
      },
      {
        "method": "GET",
        "rel": "getDefinitions",
        "href": "https://<external gateway host>/marketingWorkflow/
definitions",
        "uri": "/definitions",
        "type": "application/vnd.sas.collection",
        "itemType": "application/
vnd.sas.marketing.workflow.definition.summary"
      }
    ]
  },
  {
    "version": 1,
    "id": "b7ce5bb4-412e-4940-be2b-471abc464228",
    "name": "API Template C995441 Wed_Mar_31_23_58_55_2021",
    "description": "",
    "lastModifiedTimeStamp": "2021-04-01T04:54:34Z",
    "lastModifiedBy": "sasUser",
    "lastModifiedByName": "sasUser",
    "associatedObjectTypes": "Planner.PlanningObject",
    "workflowType": "GENERIC",
    "disabled": false,
    "filesAllowedToBeCataloged": false,
    "state": "DRAFT",
    "versionNumber": 1,
    "links": [
      {
        "method": "GET",
        "rel": "self",
        "href": "https://<external gateway host>/marketingWorkflow/
definitions/b7ce5bb4-412e-4940-be2b-471abc464228",
        "uri": "/definitions/b7ce5bb4-412e-4940-be2b-471abc464228",
        "type": "application/vnd.sas.marketing.workflow.definition"
      },
    ],
  },

```

```

        {
            "method": "GET",
            "rel": "getDefinitions",
            "href": "https://<external gateway host>/marketingWorkflow/
definitions",
            "uri": "/definitions",
            "type": "application/vnd.sas.collection",
            "itemType": "application/
vnd.sas.marketing.workflow.definition.summary"
        }
    ]
},
"limit": 20,
"version": 2
}

```

- 3 Using the workflow definition's ID from the previous step, retrieve the details for the workflow definition:

```

GET https://<external gateway host>/marketingWorkflow/definitions/
4797816a-45c4-4605-8088-439a138fee29

```

This is an example of a successful response:

```

{
    "version": 0,
    "id": "4797816a-45c4-4605-8088-439a138fee29",
    "name": "Workflow Template C995443",
    "description": "",
    "lastModifiedDate": "2021-04-01T04:54:53Z",
    "lastModifiedBy": "sasUser",
    "lastModifiedByName": "sasUser",
    "createdUserName": "sasUser",
    "createdBy": "sasUser",
    "createdByName": "sasUser",
    "createdDate": "2021-04-01T04:54:47Z",
    "model": {
        "tasks": [
            {
                "key": "idCreateArtwork",
                "defaultDurationPerAssignee": "2",
                "taskType": "GENERIC",
                "internalTaskType": "USER_TASK",
                "configuration": {
                    "showSourceItemLink": false,
                    "showWorkflowLink": false
                },
                "predecessorTaskKeys": [],
                "version": 0,
                "name": "Create Artwork",
                "assigneeType": "user",
                "assignees": [
                    {
                        "assignee": {
                            "name": "sasUser@sas.com",
                            "displayName": "sasUser@sas.com",
                            "type": "user"
                        }
                    }
                ]
            }
        ]
    }
}

```

```

    }
  },
  "multipleAssigneesSupported": true,
  "sequential": true,
  "responseConfig": {
    "overallComments": {
      "enabled": true,
      "mandatory": true
    },
    "resources": {
      "enabled": false,
      "mandatory": false,
      "files": false,
      "urls": false,
      "ciObjects": false
    }
  }
},
{
  "key": "idApproverTask",
  "taskType": "GENERIC",
  "internalTaskType": "USER_TASK",
  "configuration": {
    "showSourceItemLink": false,
    "showWorkflowLink": false
  },
  "predecessorTaskKeys": [
    "idModifyArtwork",
    "idCreateArtwork"
  ],
  "version": 0,
  "name": "Approve Artwork",
  "assigneeType": "user",
  "multipleAssigneesSupported": true,
  "sequential": true,
  "responseConfig": {
    "overallComments": {
      "enabled": true,
      "mandatory": true
    },
    "resources": {
      "enabled": false,
      "mandatory": false,
      "files": false,
      "urls": false,
      "ciObjects": false
    }
  }
},
{
  "key": "idModifyArtwork",
  "taskType": "GENERIC",
  "internalTaskType": "USER_TASK",
  "configuration": {
    "showSourceItemLink": false,

```

```

        "showWorkflowLink": false
    },
    "predecessorTaskKeys": [
        "idApproverTask",
        "idCreateArtwork"
    ],
    "version": 0,
    "name": "Modify Artwork",
    "assigneeType": "user",
    "multipleAssigneesSupported": true,
    "sequential": false,
    "responseConfig": {
        "overallComments": {
            "enabled": true,
            "mandatory": true
        },
        "resources": {
            "enabled": false,
            "mandatory": false,
            "files": false,
            "urls": false,
            "ciObjects": false
        }
    },
    },
    {
        "key": "idApproved",
        "defaultDurationPerAssignee": "2",
        "taskType": "GENERIC",
        "internalTaskType": "USER_TASK",
        "configuration": {
            "showSourceItemLink": false,
            "showWorkflowLink": false
        },
        "predecessorTaskKeys": [
            "idApproverTask",
            "idModifyArtwork",
            "idCreateArtwork"
        ],
        "version": 0,
        "name": "Approved Flow Task",
        "assigneeType": "user",
        "assignees": [
            {
                "assignee": {
                    "name": "sasUser@sas.com",
                    "displayName": "sasUser@sas.com",
                    "type": "user"
                }
            }
        ],
        "multipleAssigneesSupported": false,
        "sequential": false,
        "responseConfig": {
            "overallComments": {
                "enabled": true,

```

```

        "mandatory": true
    },
    "resources": {
        "enabled": false,
        "mandatory": false,
        "files": false,
        "urls": false,
        "ciObjects": false
    }
}
},
],
"conditionalFlows": [
    {
        "key": "gatewayToApprovedFlow",
        "expression": "${idApproverTask_STATUS == \"APPROVED\"}",
        "dataMappingFields": [
            "idApproverTask_STATUS"
        ],
        "predecessorTaskKeys": [
            "idApproverTask",
            "idModifyArtwork",
            "idCreateArtwork"
        ]
    },
    {
        "key": "gatewayToRejectedFlow",
        "expression": "${idApproverTask_STATUS == \"REJECTED\"}",
        "dataMappingFields": [
            "idApproverTask_STATUS"
        ],
        "predecessorTaskKeys": [
            "idApproverTask",
            "idModifyArtwork",
            "idCreateArtwork"
        ]
    }
],
"dataObjects": [
    {
        "dataMappingField": "idApproverTask_STATUS",
        "source": "cirest.SVC.PLANNER.PlanningObjects",
        "sourceItem": "",
        "sourceItemFieldPath": "description",
        "requiredHeaders": [],
        "details": {}
    }
],
},
"associatedObjectTypes": "Planner.PlanningObject",
"availableCategories": [],
"engineProcessDefinitionKey": "id_Workflow_template_C995443",
"workflowType": "GENERIC",
"templateAccess": {
    "publicAccessDenied": false
},

```

```

    "disabled": false,
    "filesAllowedToBeCataloged": false,
    "state": "DRAFT",
    "versionNumber": 1,
    "defaultPermissions": {
      "publicAccessDenied": false
    },
    "links": [ ... ]
  }

```

Workflow Processes

Building the Base URL for the API Calls

To successfully use the Workflow API, start with the correct URL domain.

- 1 Find the host name for the REST API.
 - a Navigate to  **Administration** ⇒ **Data and Integrations**. Click **Access Points**.
 - b Make note of the value of the **External gateway host** field. This is the root URL for any API calls that go through the external API gateway. Use the value in this field when you see the variable `<external_gateway_host>` in the documentation.

An example might use the following URL for an endpoint:

```
https://<external_gateway_host>/<endpoint>
```

- 2 Use this base URL with the API paths to make calls. For example:

```
GET https://<external_gateway_host>/marketingWorkflow/definitions
```

Example: View a List of Workflows

You can make an API call to list the workflow processes:

- 1 To authenticate with the REST API, specify your temporary JWT as the bearer token for REST API calls.

Note: Use the term “Bearer” as a prefix and add a space before the value of the JWT.

The header should be similar to this example:

Table 96 *Using a Token as the Authorization Header*

Header Attribute	Value
Authorization	Bearer eyJ0eXAiOiOGci0.DAxMGFhODEJ9.BRv_nD6v

- 2 Send a GET call to retrieve the list of workflows:

```
GET https://<external_gateway_host>/marketingWorkflow/processes
```

This is an example of a successful response:

```
{
  "name": "WorkflowProcesses",
  "start": 0,
  "limit": 20,
  "count": 250,
  "accept": "application/json",
  "links": [
    {
      "method": "string",
      "rel": "string",
      "uri": "string",
      "href": "string",
      "title": "string",
      "type": "string",
      "itemType": "string",
      "responseType": "string",
      "responseItemType": "string"
    }
  ],
  "version": 2,
  "items": [
    {
      "id": "17e38990-3640-6008-1602-01ef1h7ef319",
      "name": "Workflow for New Product Launch",
      "version": 1,
      "code": "310320202359",
      "createdUserName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
      "createdBy": "demouser",
      "createdByName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
      "lastModifiedBy": "demouser",
      "lastModifiedByName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
      "lastModifiedTimeStamp": "2024-12-01T08:44:24Z",
      "submittedBy": {},
      "submittedTimeStamp": "2024-12-01T08:44:24Z",
      "state": "SAVEASDRAFT",
      "localizedState": "Pending",
      "percentComplete": 0,
      "delayedBy": 1,
      "businessInformation": {},
      "numberOfTasks": "4",
      "workflowType": "GENERIC",
      "plannedEndTimeStamp": "2024-12-01T08:44:24Z",
      "projectedEndTimeStamp": "2024-12-07T09:23:55Z",
      "startTimeStamp": "2024-12-01T08:44:24Z",
      "endTimeStamp": "2024-12-07T09:23:55Z",
      "timelineCalculationTimeStamp": "2024-12-07T09:23:55Z",
      "links": [
        null
      ]
    }
  ]
}
```

Example: Create a Workflow

You can use the Workflows REST API to create a workflow. The same API can also be used to initiate the workflow that you created.

Query parameters are used to specify additional conditions or actions. You can use the `launchProcess` query parameter to specify whether you want to create and initiate the workflow. If the value of this parameter is passed as "true", then the new workflow process is created and launched. If the value is passed as "false", then the new workflow process is created but not launched.

Send a POST call to create a workflow process.

POST `https://<external gateway host>/marketingWorkflow/processes`

This is an example of a successful response:

```
{
  "id": "22e935e2-460d-4527-987e-a9e6ee014c4a",
  "name": "New Product Launch Workflow",
  "version": 1,
  "code": "200420201208",
  "description": "Description for new product launch",
  "ownerName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
  "createdUserName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
  "createdBy": "demouser",
  "createdByName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
  "creationTimeStamp": "2024-12-01T08:44:24Z",
  "lastModifiedBy": "demouser",
  "lastModifiedByName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
  "lastModifiedTimeStamp": "2024-12-01T08:44:24Z",
  "processDefinitionId": "f4075806-eed6-45c9-a931-f7ce34fa0da6",
  "submittedTimeStamp": "2024-12-01T08:44:24Z",
  "state": "INPROGRESS",
  "localizedState": "Pending",
  "comment": "some comment",
  "businessInformation": {},
  "numberOfTasks": "4",
  "workflowType": "GENERIC",
  "category": "category",
  "plannedEndTimeStamp": "2024-12-01T08:44:24Z",
  "projectedEndTimeStamp": "2024-12-07T09:23:55Z",
  "startTimeStamp": "2024-12-01T08:44:24Z",
  "endTimeStamp": "2024-12-07T09:23:55Z",
  "timelineCalculationTimeStamp": "2024-12-07T09:23:55Z",
  "percentComplete": 70,
  "delayedBy": 0,
  "submittedBy": {},
  "mappedGroupMembers": [
    {
      "name": "demouser",
      "displayName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
      "type": "USER",
      "members": [
        {}
      ]
    }
  ]
},
```

```

"tasks": [
  {
    "id": "ea0f7d8c-c925-4b0d-8078-1ed9b00f4064",
    "name": "Approve Artwork For Generated Artwork",
    "version": 1,
    "description": "Description for task",
    "initiatorUserName": "fSsXQ7DNCN15pKyW8FF7f4nMVqvsoyh2",
    "lastModifiedTimeStamp": "2024-12-01T08:44:24Z",
    "key": "idApproverTask",
    "processId": "45c3f71c-1580-4e4d-ac21-3accdc9abb6e",
    "processName": "Promotion Revision Workflow",
    "processCode": "200320200908",
    "workflowType": "GENERIC",
    "sequential": false,
    "businessInformation": {},
    "processDefinitionId": "72563198-889a-4e86-9a56-e8bca63f04a0",
    "taskTypeForLink": "MyTask",
    "taskType": "ITEMAPPROVAL",
    "internalTaskType": "USER_TASK",
    "versionNumber": 1,
    "isLatest": true,
    "locallyUpdated": true,
    "processSubmittedBy": {},
    "state": "NOT_INITIATED",
    "localizedState": "Not initiated",
    "multipleAssigneesSupported": false,
    "instruction": "some instruction",
    "projectedStartTimeStamp": "2024-12-01T08:44:24Z",
    "projectedEndTimeStamp": "2024-12-07T09:23:55Z",
    "startedTimeStamp": "2024-12-01T08:44:24Z",
    "endTimeStamp": "2024-12-07T09:23:55Z",
    "dueTimeStamp": "2024-12-01T08:44:24Z",
    "percentComplete": 0,
    "delayedBy": 0,
    "defaultDurationPerAssignee": "3",
    "links": [
      null
    ]
  }
],
"customAttributes": {},
"links": [
  null
]
}

```

Developer Resources By API Collection

Use the following links to access the developer documentation for each REST API:

- [Container Image Management API \(/marketingContainerImage\)](#)
- [Content Request API](#)

- [Copy Item API \(/marketingPromotion\)](#)
- [Digital Assets API \(/marketingDigitalAssets\)](#)
- [Marketing Administration API \(/marketingAdmin\)](#)
- [Marketing Audience API \(/marketingAudience\)](#)
- [Marketing Data API \(/marketingData\)](#)
- [Marketing Design API \(/marketingDesign\)](#)
- [Marketing Execution API \(/marketingExecution\)](#)
- [Marketing Gateway API \(/marketingGateway\)](#)
- [Plan API \(/marketingPlan\)](#)
- [Server-Side Events API](#)
- [System for Cross-Domain Identity Management \(SCIM\) API \(/scim\)](#)
- [Workflow API \(/marketingWorkflow\)](#)