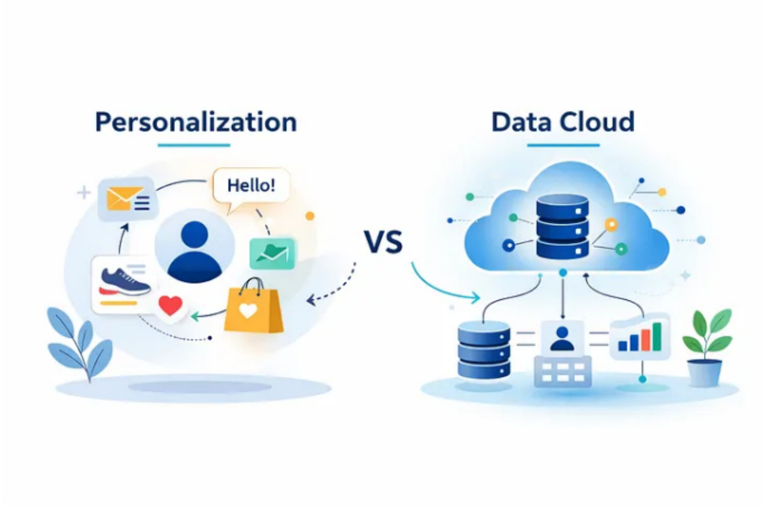
Developer's Guide : Personalization (Interaction Studio) vs. Data Cloud (Web SDK) Sitemap

CloudCove.ai Follow 9 min read · 6 days ago



1. Introduction to Sitemap Integration and Configuration

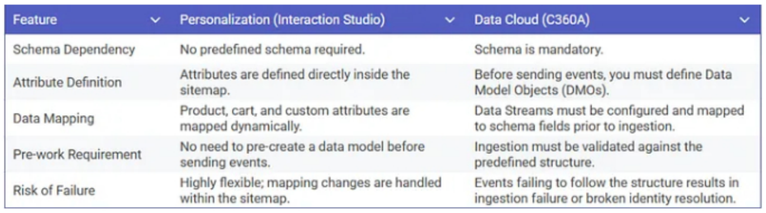
Salesforce Personalization is used for **real-time engagement**, using a flexible, schema-less approach to define attributes directly within the sitemap for immediate, experiences.

Conversely, Salesforce Data Cloud is built for data streams that requires a mandatory, predefined schema where **Data Lake Objects (DLOs)** must be defined and data must be mapped before use.

Both use the same Salesforce Interactions SDK.

2. The Architectural Divide: Schema Requirements

The most significant architectural difference between these two implementations is how they handle data structures. In Personalization, the sitemap creates the real-time events. In Data Cloud, the sitemap must reflect a defined event based on schema.



Feature	Personalization (Interaction Studio)	Data Cloud (C360A)
Schema Dependency	No predefined schema required.	Schema is mandatory.
Attribute Definition	Attributes are defined directly inside the sitemap.	Before sending events, you must define Data Model Objects (DMOs).
Data Mapping	Product, cart, and custom attributes are mapped dynamically.	Data Streams must be configured and mapped to schema fields prior to ingestion.
Pre-work Requirement	No need to pre-create a data model before sending events.	Ingestion must be validated against the predefined structure.
Risk of Failure	Highly flexible; mapping changes are handled within the sitemap.	Events failing to follow the structure results in ingestion failure or broken identity resolution.

Note: *This difference is a balance between flexibility and structure.*

Personalization does not require a strict schema, so it can be implemented quickly and easily updated or tested, making it ideal for fast changes and real-time personalization. Data Cloud requires a predefined schema, which takes more effort initially, but this structured approach allows it to connect data from multiple sources and accurately combine it into a single unified customer profile, enabling a complete 360-degree view of the customer.

3. Sitemap Skeleton

The sitemap skeleton defines how the SDK initializes and how configurations are passed to the platform.

Personalization Sitemap Skeleton

```
SalesforceInteractions.init({
    cookieDomain: "company.com",
}).then(() => {
    const sitemapConfig = {
        global: {},
        pageTypes: [],
    };
    SalesforceInteractions.initSitemap(sitemapConfig); // Initializes the Sitema
});
```

Personalization Sitemap Skeleton defines cookie domain, global events, page types, and user attributes.

Note: *The manual structure in Personalization is designed to provide flexibility. It allows developers and architects to define or update page logic whenever needed, without depending on a fixed structure.*

Data Cloud Sitemap Skeleton

```
SalesforceInteractions.init().then(() => {
    SalesforceInteractions.initSitemap({
        global: { ... },
        pageTypeDefault: { ... },
        pageTypes: [...]
    })
}))
```

Data Cloud sitemap is same as Personalization sitemap. In that we must declare event types and profile category.

Note: *The* ***pageTypeDefault*** *in the Data Cloud skeleton acts as a safety backup. It ensures that even if a user visits a page that is new or not properly mapped, the system still captures the interaction data.*

4. Technical Deep Dive: Custom Events

Personalization Custom Event
The Salesforce Interactions SDK and the Personalization module feature several built-in CustomEvents that you can use to bind listeners specific to the Web SDK.

List of Custom Events:
SalesforceInteractions.mcis.CustomEvents

```
{
  OnEventResponse: "mcis:onEventResponse",
  OnBeforeEventSend: "mcis:onBeforeEventSend",
  OnStatSend: "mcis:onStatSend",
  OnTemplateDisplayEnd: "mcis:onTemplateDisplayEnd",
  OnInit: "mcis:onInit"
}
```

SalesforceInteractions.CustomEvents

```
{
  OnBeforeEventSend: "interactions:onBeforeEventSend",
  OnClearPersistedIdentities: "interactions:onClearPersistedIdentities",
  OnConsentRevoke: "interactions:onConsentRevoke",
  OnException: "interactions:onException",
  OnInit: "interactions:onInit",
  OnInitSitemap: "interactions:onInitSitemap",
  OnPageMatchStatusUpdated: "interactions:onPageMatchStatusUpdated",
  OnResetAnonymousId: "interactions:onResetAnonymousId",
  OnSetAnonymousId: "interactions:onSetAnonymousId",
  OnShutDown: "interactions:onShutDown",
  OnClearCookie: "interactions:onClearCookie"
}
```

Example Usage:

```
document.addEventListener(SalesforceInteractions.mcis.CustomEvents.OnStatSend, (
  console.log(domEvent.detail.campaignStat.stat);
});
const stat = {
  experienceId: "test",
  control: false,
  stat: "Impression",
};
SalesforceInteractions.mcis.sendStat({ campaignStats: [stat] });
```

Data Cloud Custom Event

Define the Schema in Data 360: Before sending the event, you must modify your Web Connector schema in Data 360 to define the event's unique name and its expected custom fields.

```
{
  "records": [
    {
      "developerName": "myCustomEvent",
      "masterLabel": "My Custom Event",
      "category": "Engagement",
      "externalDataTranFields": [
        {
          "masterLabel": "My Custom Field",
          "dataType": "Text",
          "developerName": "myCustomField",
          "isDataRequired": true
        },
        {
          "masterLabel": "My Custom Num Attribute",
          "dataType": "Number",
          "developerName": "attributesMyCustomNum",
          "isDataRequired": true
        },
        {
          "masterLabel": "My Custom Text Attribute",
          "dataType": "Text",
          "developerName": "attributesMyCustomText",
          "isDataRequired": true
        },
        },
        {
          "masterLabel": "eventId",
          "dataType": "Text",
          "developerName": "eventId",
          "isDataRequired": true,
          "primaryIndexOrder": 1
        },
        {
          "masterLabel": "category",
          "dataType": "Text",
          "developerName": "category",
          "isDataRequired": true
        },
        {
          "masterLabel": "dateTime",
          "dataType": "DateTime",
          "developerName": "dateTime",
          "isDataRequired": true
        },
        {
          "masterLabel": "deviceId",
          "dataType": "Text",
          "developerName": "deviceId",
          "isDataRequired": true
        },
        {
          "masterLabel": "eventType",
          "dataType": "Text",
          "developerName": "eventType",
          "isDataRequired": true
        },
        {
          "masterLabel": "sessionId",
          "dataType": "Text",
          "developerName": "sessionId",
          "isDataRequired": true
        }
      ]
    }
  ]
}
```

Capture the Event with the SDK: To capture and send the custom event from your website, use the `SalesforceInteractions.sendEvent()` function and specify the `eventType` or `name` field, and required fields that aren't set via Automatic Population.

```
SalesforceInteractions.sendEvent({
  interaction: {
    name: "myCustomEvent",
    eventType: "myCustomEvent",
    myCustomField: "some value",
    attributes: {
      myCustomNum: 1234,
      myCustomText: "abcd",
    },
  },
});
```

Event JSON Payload: The SDK combines your data with automatically populated fields and transforms the names to match the Web Connector schema.

```
{
  "events": [
    {
      "eventType": "myCustomEvent",
      "myCustomField": "some value",
      "attributesMyCustomNum": 1234,
      "attributesMyCustomText": "abcd",
      "eventId": "100d237b-e464-44df-9556-7481bbf52685",
      "dateTime": "2022-10-18T18:25:26.680Z",
      "sessionId": "6b6c33bfcfee93f4",
      "deviceId": "6b6c33bfcfee93f4",
      "interactionName": "myCustomEvent",
      "sourceUrl": "https://{my site}"
```

```
        "sourceUrlReferrer": "https://{my site}",
        "sourceChannel": "Web",
        "sourcePageType": "test",
        "category": "Engagement"
      }
    ]
  }
```

Note: *Observe that the SDK automatically adds* ***sessionId*** *and* ***deviceId*** *to every event. These fields help identify the user's browser session and device uniquely.*

5. Core Architectural difference: Data Classification and Events Validation

In Personalization we can see real-time engagement in personalization event stream, where we can check event data, user profile data(emailAddress, FirstName, LastName, etc) and product catalog data.

There is **no need to define structural separation** between behavioral and profile data at the data model level.

How Data Is Stored

· Page views, product views, clicks → Stored as events
· User attributes (email, loyalty tier, customer Id) → Attached to user profile dynamically
· Attributes can be injected at runtime using onActionEvent

User attribute data and view events data used in below:
· Real-time campaign evaluation
· Dynamic personalization
· Runtime enrichment

Data Cloud Web SDK:

Data Cloud follows a **strict classification-based architecture.**
All incoming data is divided into categories:
Engagement Data
Profile Data

Engagement Data (Behavioral)
As an engagement data we can capture page view, product view, click events, add to cart event and custom behavioral interactions
These engagement data is automatically stored under:
Data Model Objects (DMOs)

Characteristics:
The main characteristic of Engagement data is to track and capture user behavioral data
Behavioral data → Saved in **Engagement** by default.

Profile Data (User Attribute)
As a profile data we can capture emailAddress, CRM ID, customerId,

FirstName, LastName, phoneNumber, etc
These profile data is stored in:
Profile Data Model Objects

Characteristics:
The main characteristic of Profile data is to capture user information and it is used in identity resolution which supports unified customer profile
User data → Saved in **Profile**.

6. Comparison of Beacon Tags

Every sitemap implementation begins by adding a beacon tag into your website. Both Data Cloud and Personalization use the same Salesforce Interactions SDK, but each beacon connects to a different system. This helps load the right features. Personalization focuses on tracking user behavior in real time, while Data Cloud focuses on collecting and combining customer data from different sources.

Personalization Beacon Tag The Personalization sitemap uses the **Evergage JavaScript SDK**. This script loads the tracking library that captures user behavior and page interactions, sending data to the Personalization engine for real-time targeting and recommendations.

Personalization beacon → Real-time engagement & targeting

```
<script src="https://cdn.c360a.salesforce.com/beacon/example.com/scripts/c360a.m
```

Data 360 Beacon Tag The Data Cloud sitemap uses the **c360a (Data 360) beacon**. This script connects your website to Salesforce Data Cloud, enabling event ingestion, identity resolution, and unified customer profile creation.

Data 360 beacon → Data collection & customer unification

```
<script src="https://<accountName>.<instance>.evergage.com/scripts/evergage.min.
```

Note: *Even though the original outline instructions were mixed up, I have focused on keeping the technical information correct based on the actual system behavior. This ensures that the Evergage SDK is properly explained as the component responsible for real-time user engagement, and the C360A beacon is correctly associated with collecting data for Data Cloud and building unified customer profiles.*

7. Debugging with the Salesforce Interactions SDK Launcher

Validation is the final hurdle in any implementation. The Salesforce Interactions SDK Launcher is our primary tool, but the "Product Type" selection acts as the critical pivot point for how we interpret the results.

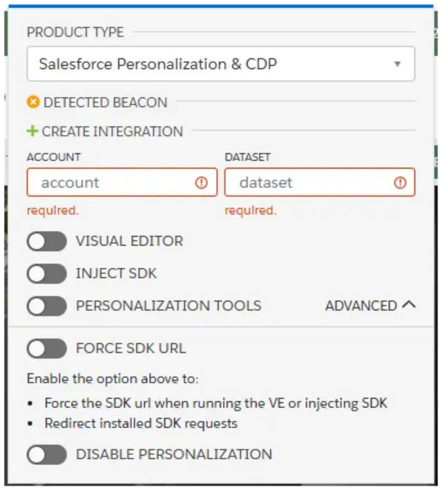
Personalization Sitemap Debug:

When selecting **Product Type → Salesforce Personalization & CDP**, the launcher focuses on real-time behavior and visual triggers.

What You Can Test:
You can test whether the beacon is working correctly or beacon is injected properly on the website. By SDK launcher we can test sitemap, in that we can check page type, content zones, listners are matching correctly or not. We can also create web template and web campaigns through SDK launcher.

The initial steps of the SDK launcher require adding the account name and dataset name. Without this, we cannot proceed with further actions. If the beacon is not present on the website but we still want to test the sitemap or web campaign, we need to force SDK integration. By doing this, we can inject the beacon tag into the specific website.

When we enable visual editor we can see sitemap, web template, web campaign and content zone options.



PRODUCT TYPE
Salesforce Personalization & CDP
DETECTED BEACON
+ CREATE INTEGRATION

ACCOUNT	DATASET
account	dataset
required.	required.

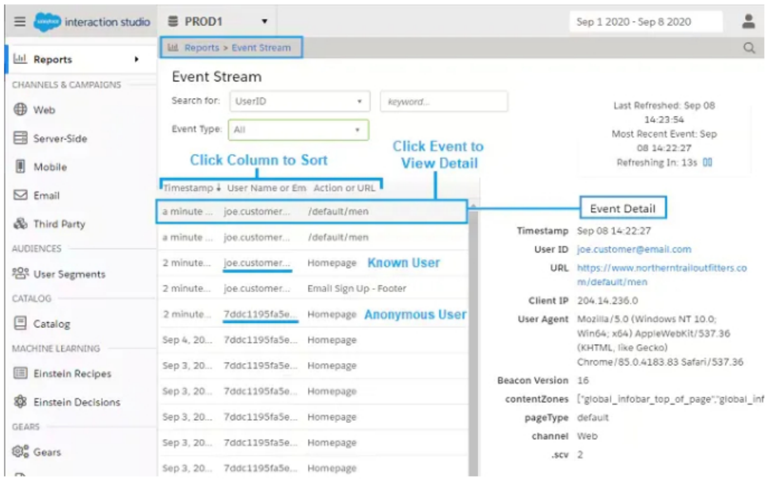
VISUAL EDITOR
INJECT SDK
PERSONALIZATION TOOLS
ADVANCED
FORCE SDK URL

Enable the option above to:

- Force the SDK url when running the VE or injecting SDK
- Redirect installed SDK requests

DISABLE PERSONALIZATION

When we debug sitemap through SDK launcher we can see the specific event and user data in Personalization event stream and in user profile.



Event Stream

Search for: UserID keyword...
Event Type: All

Last Refreshed: Sep 08 14:23:54
Most Recent Event: Sep 08 14:22:27
Refreshing In: 13s

Timestamp	User Name or Em	Action or URL
a minute ...	joe.customer...	/default/men
a minute ...	joe.customer...	/default/men
2 minute...	joe.customer...	Homepage
2 minute...	joe.customer...	Email Sign Up - Footer
2 minute...	7ddc1195fa5e...	Homepage
Sep 4, 20...	7ddc1195fa5e...	Homepage
Sep 3, 20...	7ddc1195fa5e...	Homepage
Sep 3, 20...	7ddc1195fa5e...	Homepage
Sep 3, 20...	7ddc1195fa5e...	Homepage
Sep 3, 20...	7ddc1195fa5e...	Homepage
Sep 3, 20...	7ddc1195fa5e...	Homepage

Event Detail

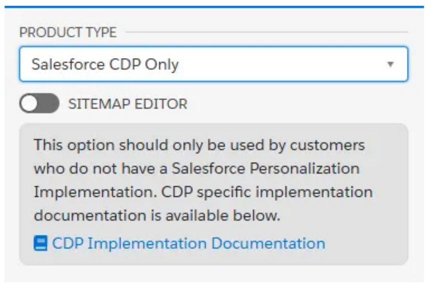
Timestamp	Sep 08 14:22:27
User ID	joe.customer@email.com
URL	https://www.northerntrailoutfitters.com/default/men
Client IP	204.14.236.0
User Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36
Beacon Version	16
contentZones	["global_infobar_top_of_page","global_inf
pageType	default
channel	Web
.scv	2

Data Cloud Sitemap Debug:

When selecting **Product Type → Salesforce CDP Only**, the launcher pivots to validate the integrity of the data being prepared for the Data Cloud ingestion engine.

What You Can Test:

You can test whether the SDK is properly loaded on the website, ensure that events are being sent in the correct format, confirm that user identity information is captured correctly, check that the data matches the required schema, and make sure the data is successfully sent and received in the data stream.


PRODUCT TYPE

Salesforce CDP Only

SITEMAP EDITOR

This option should only be used by customers who do not have a Salesforce Personalization Implementation. CDP specific implementation documentation is available below.

CDP Implementation Documentation

In Data Cloud we can validate the data sent by sitemap in data explorer or query data.

8. Avoid Runtime Mutation : Understanding the Role of onActionEvent

In **Personalization,** When a user does something on your website like viewing a product, logging in, adding to cart or making a purchase, an **event** is sent to **Marketing Cloud Personalization.** Before that event is officially sent, **onActionEvent** gives you a chance to add missing data like user data firstname, lastname , emailaddress that is why these action adds to specific user and without **onActionEvent** we cannot merge anonymous user to known user.

When a User Logs In After Browsing Anonymously
The Problem:
Most users browse your website without logging in.
So the system tracks them as: **Anonymous User**
Later, when they log in, the system sees: **Known User**

How onActionEvent Helps ?
When the user logs in, **onActionEvent** can attach their email or customer ID to the event before sending it.
This helps Salesforce understand: this anonymous visitor and this logged-in user are the same person.

Why This Is Important ?
· Past browsing history is preserved
· Profiles are merged

Personalization becomes accurate immediately after login

Example Code: Identity Enrichment

```
onActionEvent: (actionEvent) => {
  const email = SalesforceInteractions.mcis.getValueFromNestedObject("window._et
  if (email) {
    actionEvent.user = actionEvent.user || {};
    actionEvent.user.identities = actionEvent.user.identities || {};
    actionEvent.user.identities.emailAddress = email;
    actionEvent.user.attributes.firstName = firstName;
    actionEvent.user.attributes.lastName = lastName;
  }
```

```
    return actionEvent;
  },
```

Data Cloud In one implementation, we used onActionEvent globally to attach an email address and force the event type to "identity".

Example Code:

```
  onActionEvent: (actionEvent) => {
    const email = SalesforceInteractions.mcis.getValueFromNestedObject("window._et
    if (email) {
      actionEvent.user = actionEvent.user || {};
      actionEvent.user.attributes = actionEvent.user.identities || {};
      actionEvent.user.attributes.emailAddress = email;
      actionEvent.user.attributes.eventType = "identity"; // Fatal Error: Forcing
    }
    return actionEvent;
  },
```

What Went Wrong ?
Because the logic was placed globally, it ran for every event such as page view, product view, add to cart, purchase, and custom events, causing the identity event to be sent repeatedly along with each event instead of being sent just once.

The Result: This created massive duplicates in Profile DMOs, triggered validation errors, and caused ingestion failures. Data Cloud requires identity data to be sent intentionally and structured correctly, not injected casually across immutable, time-series **Engagement Data**.

9. Conclusion: Architectural Purposes

Even though Salesforce Personalization and Data Cloud share the same Interactions SDK foundation, they use sitemaps for distinct architectural goals. In Personalization, the sitemap is focused on **real-time action**, handling page detection, triggering active campaigns, and defining content zones to tailor the user experience instantly. Conversely, Data Cloud uses the sitemap for **structured data ingestion with default frequency of 15 minutes,** to further leverage this data for use cases it depends on **Identity Resolution Ruleset,** ensuring events are formatted to match strict schemas and support identity resolution for building unified customer profiles.

P.S- To leverage the full real-time capabilities of Data Cloud and enable real-time use cases, a separate license is required.

CloudCove.ai
https://www.linkedin.com/company/cloudcove-ai

Salesforce Personalization Data Cloud Saleforce Marketing Cloud Sitemap

2

Written by CloudCove.ai

4 followers · 3 following

Follow