Self-Service Flow Actions Service Provider Interface

https://github.com/adobe/Marketo-SSFS-Service-Provider-Interface

https://github.com/adobe/aio-compute-formula

Self-service Flow Actions is a framework for creating and publishing HTTP APIs for consumption by Marketo Smart Campaigns as flow actions. The accompanying OpenAPI/Swagger document is a Service-Provider Interface describing how an API must be implemented for automatic integration to Marketo instances. Implementation of an API requires at least 3 and as many as 7 endpoints, definition of an authentication schema, and the components and schemas required for implementation.

Changelog

6/4/22 Release (1.0.0)

- Schema version now 1.0.0
- Services using Beta schemas will have continued support for at least one year, but services intended for long term use should update x-schema-version to 1.0.0

Changes for 5/9/22 Release (3.1)

- Support added for initiating installation via redirect. A link formatted like https://app.marketo.com/custAdmin/addServiceProvider?url=<url formatted location of your swagger definition> will initiate installation to a Marketo instance for authenticated users.
- Support added for OAuth2 Authentication Code Grant, AKA Three-Legged OAuth, to support simplified and secure authentication for users.

Changes from 3.0

- Schema and tests added for validating OpenAPI definitions.
- Added full authoring section.
- icon and provider instructions booleans in serviceDefinition have been removed. These endpoints will be inferred based on whether they are included as paths in your API definition.

Changes from 0.2.3

- Installation is now initiated with Swagger API definition rather than serviceDefinition endpoint.
- Removed auth, support contact, settings from serviceDefinition and moved them to Swagger.
    - Authentication should be defined using security and securitySchemes.
    - Use x-fields to define non-standard names or patterns.
- Use x-schemaVersion to indicate version of CFA-Swagger.yaml used to define your API.

Authentication

Currently, Basic, API-Key, OAuth2 Client Credentials, Refresh Token and Authorization Code grant types are supported. Authentication type is set in your swagger definition using the securitySchemes object.

Basic

Setting the authType to 'basic' will prompt end users for a username and password during service configuration. If your service does not use the 'realm' component of basic authentication as defined in RFC 7235, then you should also set realmRequired to 'false.' During invocation, Marketo will encode the credentials as defined by the RFC and send them in the Authorization header.

API Key

API key authentication is supported in either header or query params.

OAuth2

Client Credentials, Authentication Code, and Refresh Token grant types are supported. One of Client Credentials or Authentication code must be used, while Refresh Token is optionally supported in addition to one of these. If multiple OAuth2 flows are described in your API definition, the authorizationCode flow will be preferred over clientCredentials.

Authoring

Your API definition must conform both to the OpenAPI 3 specification and the included schema.yaml. This section will only discuss requirements beyond the OpenAPI specification. More information on OpenAPI requirements can be found here. The example API definition provides extensive examples and descriptions of the metadata described in this section.

info

Your info section must include:

- x-providerName: A *string* which is used as the name of your service provider when installed in a Marketo instance.
- x-schemaVersion: A *string* which must match a version of the schema used to create it. This corresponds to this repository's version in package.json.
- x-supportContact: A *string* which must be either an email address or URL which users can use to access support for your service.

If servers is not set in your API definition, Marketo will infer your base path as whatever precedes the URL of your OpenAPI/Swagger file. For example, if you have https://www.example.com/api/swagger.json, but have not set servers in the file, https://www.example.com/api/ will be considered the root of your API paths.

Security

Your security section must use one of the sample schemes, apiKey, oauth2, or basic.

Paths

/getServiceDefinition

This endpoint describes most of the configuration required to implement a service, includes links to other endpoints, describes the chosen authentication scheme, and describes the lead, activity, and contextual data required by the service to operate.

Your service definition requires:

- apiName: A *string* which is the default identifier for service and activity. Users installing multiple services with the same apiName will be prompted to resolve collision by inputting a custom name during installation. Values 'success', 'reason', and 'errorCode' are always included in activityData and may not be declared here, see '#components/schemas/callbackData'.
- i18n: An *object* (serviceI18nObject) used to provide localized user-friendly strings used in the UI.
- primaryAttribute: A *string* which is the API name of the attribute that describes the primary asset. This must match an attribute from the flow attribute list and *must not* match an attribute from the callback attribute list.
- invocationPayloadDef: An *object* used to describe user inputs, and data from leads and execution context, which is sent by Marketo to your service upon invocation.
- callbackPayloadDef: An *object* used to describe data which will be returned by your service in the callback to Marketo.

invocationPayloadDef

- globalAttributes: A *list* of attributes (invocationAttributeObjects) describing expected global user inputs. Global attributes can be set during installation or from the Service Provider admin menu. Global attributes will be included in every invocation if set.
- flowAttributes: A *list* of attributes (invocationAttributeObjects) describing expected flow step inputs. Flow attributes are set for each individual instance of a flow step and are sent per-lead in the flowStepContext object.
- fields: A *list* of field mappings (invocationFieldMapping) needed for invocation. Fields which are mapped in Marketo are sent in the leadContext object. If userDrivenMapping is 'true', the contents of this array will be ignored.
- headers: A *list* of headers (headerAttributeObject) to be included in invocations of /async. Headers can be set during installation or from the Service Provider admin menu. Like global attributes, headers will be included in every invocation if set.

- userDrivenMapping: A *boolean* which indicates whether the service will provide a pre-defined list of mappings for outgoing fields. If 'true', 'fields' will be ignored, and mappings must be added manually by users in the UI, see User Driven Mappings.
- programContext: A *boolean* indicating whether to send program context on invocation.
- campaignContext: A *boolean* indicating whether to send campaign context on invocation.
- triggerContext: A *boolean* indicating whether to send trigger context on invocation.
- programMemberContext: A *boolean* indicating whether to send program member context on invocation.
- subscriptionContext: A *boolean* indicating whether to send subscription context on invocation.
- myTokenContext: A *list* of strings used to indicate which My Token values from the executing context should be sent upon invocation. The list should be formatted without brackets or prefix, e.g., a token from the UI called "{{my.Event Date}}" would be requested as "Event Date".

Flow and Global Parameters

Aside from field mappings, Flow and Global parameters are the primary means of parameterization when invoking a service. If parameters have suggested values, then the picklistUrl attribute should be populated with your /getPicklist URL. If this parameter can only accept a fixed set of values, then it should have both a picklistUrl and have enforcePicklistSelect set to true.

Flow parameters are assigned at the individual flow step level, meaning that these parameters may have completely different values from one campaign to another. In our event-registration example, we would need to define an "Event" Flow parameter as a string to select the event to register for. In most cases, it's easier for services to deal with IDs and users to deal with Names, so for cases like this, you should consider configuring the parameter as a picklist so that you can offer the Event Name to the user but receive the submitted ID value. See /getPicklist for more information.

Flow Parameters and Activity Attributes must not have any overlapping field names.

Global parameters are assigned at the service level by an admin user. Global params are submitted with every invocation request. In our Lookup Table use case example, "Directory" would be an example of a global parameter, where in order to provide a reduced picklist of tables to the end user, the admin would give the value of the directory where the relevant lookup tables for their instance live on the service-side.

Invocation attributes require:

- apiName: A *string* used as the key for the attribute during invocation.
- i18n: An *object* used to provide localized, user-friendly attribute names in the UI. Attributes are of the type attributeI18nObject. en_US is always required, while other localizations may be provided using four-letter language/locale keys, e.g., 'ca_FR'.
- dataType: A *string* matching a fieldType. Any of: boolean, integer, date, datetime, email, float, score, string, url, or text.

Context Data

Various types of contexts may be requested to aid your service in processing lead data. The following are boolean fields used to indicate your service should send context for that object if available:

- programContext
- campaignContext
- triggerContext
- programMemberContext
- subscriptionContext

You may also send My Token context by giving an array of strings in the myTokenContext parameter:

```
"myTokenContext": ["Event Date", "Event Address"]
```

callbackPayloadDefObject

Your callbackPayloadDef is used to define what data may be returned by the service in the callback, and how that data can be mapped back to lead and activity records.

- attributes defines result fields for your callback and will be logged as an activity related to the corresponding lead when data is sent.
- fields defines which fields you want to write lead data to. Fields defined here may be mapped by admin users to send data to the correct fields for a particular Marketo instance.
- userDrivenMapping: A *boolean* which indicates whether the service will provide a pre-defined list of mappings for incoming fields. If 'true', 'fields' will be ignored, and mappings must be added manually by users in the UI, see User Driven

Mappings, otherwise incoming fields will use Service-Driven Mappings.

Field Mappings

In order for lead data to be sent to or received from a service, those fields must be mapped to an existing Marketo field. Field mappings have two types, outgoing and incoming (relative to invocation by Marketo). Outgoing fields are sent by Marketo to the service during invocation, while incoming fields are received by Marketo through the callback and have their values written back to the lead record. There are also two usage modes for field mappings: Service-Driven Mappings for services that have a fixed and predetermined set of person-fields to complete data processing, like an event registration service, and User-Driven Mappings for services that have generic arguments, like a service for looking up data from tables uploaded by users.

Fields that have been mapped, whether user or service-driven, are sent to the service when refreshing picklist choices in the fieldMappingContext object, so that mapped fields may be used to generate choices. See Picklists.

Service-Driven Mappings

If your service requires a fixed set of inputs to process a record, then using Service-Driven Mappings is likely the correct choice for you. Take an example use case, registering for an event. A typical registration will require Full Name, Contact Info (Phone and/or Email), and Job Title. In this case, your invocationPayloadDef.fields might look like this:

JSON

```
[
  {
    "required": true,
    "serviceAttribute": "JobTitle",
    "suggestedMarketoAttribute": "title",
    "dataType": "string"
  },
  {
    "required": true,
    "serviceAttribute": "Email",
    "suggestedMarketoAttribute": "email",
    "dataType": "email"
  },
  {
    "required": true,
    "serviceAttribute": "Phone",
    "suggestedMarketoAttribute": "phone",
    "dataType": "string"
  },
  {
    "required": true,
    "serviceAttribute": "FirstName",
    "suggestedMarketoAttribute": "firstName",
    "dataType": "string"
  },
  {
    "required": true,
    "serviceAttribute": "LastName",
    "suggestedMarketoAttribute": "lastName",
    "dataType": "string"
  }
]
```

In this example, when onboarding, JobTitle will default to mapping to the field with the REST API Name 'title.' While this is likely ideal for most cases, some Marketo subscriptions may be configured to use the Job Title field differently than the service provider expects. If a subscription uses this field to hold info on Job Function or Job Level, then an admin might choose to map to a field which has more appropriate data based on their own instance configuration. Service-driven mappings work in the same way for the callback as they do for invocation, except lead-fields in the callback may not be required, so Admins may always choose to leave callback fields unmapped.

User-Driven Mappings

If your service has a flexible set of inputs and outputs, then user-driven mappings are likely the best choice for your service. Using a lookup table flow step as an example, if we have a country code lookup table where we need to send a 'country' field and receive a 'countryCode' field, then an admin will need to manually add those fields during onboarding.

Activity Attributes

Activity attributes define the data that you can send back and write to an activity in the 'attributes' of your callbackPayloadDef in your service definition. The full attribute list of your activity is the combination of your activity attributes and flow attributes defined in invocationPayloadDef. Activities in Marketo serve two primary purposes: driving triggered events and recording an event related to a person. **You may not use the names success, reason, or errorCode** as these are reserved created for all SSFS activity types and can be written to in the selfServiceFlowComplete Callback. When written, activities will log both the values submitted in the callback and the parameter values of the executed flow step choice.

Flow Parameters and Activity Attributes must not have any overlapping field names.

The **primaryAttribute** field must also be a flow parameter.

/async

This endpoint is invoked by Marketo when the flow action is invoked by a Marketo Smart Campaign. Marketo sends lead data, execution context, flow parameters, and global parameters to this endpoint, as well as a callback URL and one-time-use authentication token, so that the service can return data via the callback. The invoker expects the service to return a 201 upon successful acceptance of the request. Synchronous invocation is not supported.

selfServiceFlowComplete Callback

When processing of the invocation request has been completed, lead and activity data are returned via callback. Data must be passed back to lead fields and activity attributes in the same manner as described by the service definition.

When Data Value Change activities are recorded as the result of a callback, the "Source" and "Reason" attributes will be populated with the following data:

- Source: "{Service Name} ({Id})"
- Reason: "Smart Campaign: {Id}, Step Seq ID: {Id}"

Default Values

Lead and activity data can have default values set through the callback. This can reduce the amount of data which needs to be sent over the wire and can simplify mapping data back to Marketo if appropriate for your service.

Errors

Chunk-level errors not reported by the HTTP response to invocation should be represented by errorCode and errorMessage in the callback payload. errorCode is used to classify the error in Marketo service logs, and each instance will increment the error reporting count for the day. errorMessage will be recorded to the logged event. Error codes for individual activities and chunk-level failures should not have overlapping names to avoid complications in statistical reporting.

callbackData

callbackData is where record-specific values for the person and activity are written in leadData and activityData respectively. Each callbackData must have a leadData with an _id property, or it will be recorded as unsuccessful. Other properties must be defined in the serviceDefinition and mapped by an admin of the invoking instance in order to be written. In order to correctly report on failures to execute a job successfully for an individual record, in activityData, you should set success to false and populate errorCode with a string classifying the reason for failure, e.g., LOOKUP_VALUE_NOT_FOUND, and reason with a detailed message specific to the failure, e.g., "No value found for search parameters, Key: country Value: Cascadia." Error codes for individual activities and chunk-level failures should not have overlapping names to avoid complications in statistical reporting. If a record...

/status

Status and health endpoint that the service may use to provide Informational, Warning, or Error notifications outside of the invocation/callback workflow. This endpoint is invoked by Marketo in

Tyron Debugging Notes

Unexpected Exception: SocketTimeoutException -- Read timed out

This occurs because we are not sending the 202 upon receipt of the payload from Marketo

We need to send the 202 response right away to Marketo and then do the rest of our processing in the background and then once finished then send the callback object to update the leads