API Workflow: Customer Signup and Login for S2S Implementations

Table of Contents

Summary

SessionM provides several ways for a customer to log into or sign up for a loyalty program on a client site. Signup and login logic can be implemented directly for a client site or indirectly via a social media site. The goal is to ensure that a customer profile exists on the SessionM Platform and can be retrieved for updates. This workflow documents the steps that enable these processes for a server-to-server (S2S) implementation.

One of the primary characteristics of the S2S flow is that it begins without knowing whether or not the customer has a profile on the platform. So the first step is to perform a search that can confirm this. For example, the system sends the customer with a dialog soliciting their email address. Then, based on the search results, the system can present the appropriate next step: a login page for existing customers or a signup page for new customers. Once the customer is logged in or signed up, the workflow then uses the customer ID to retrieve the customer profile.

Use Cases

These use cases are typical for customers signing up or logging in via a web site or mobile app. Once signed in or logged in, they provide personal data that can be used to create a customer profile on the SessionM Platform:

- Customer signs up on a client site using an email and password; data becomes inputs for a new customer profile.
- Customer signs up via a social media site; data becomes inputs for a new customer profile.
- Customer logs in on a client site using an email and password or with an access token from a social media site.
- ID is obtained for S2S transactions.
- Customer retrieves and updates their profile with new data using an ID. Subsequent activities for logged in or signed in customers rely on other SessionM APIs such as those that manage campaigns or offers.

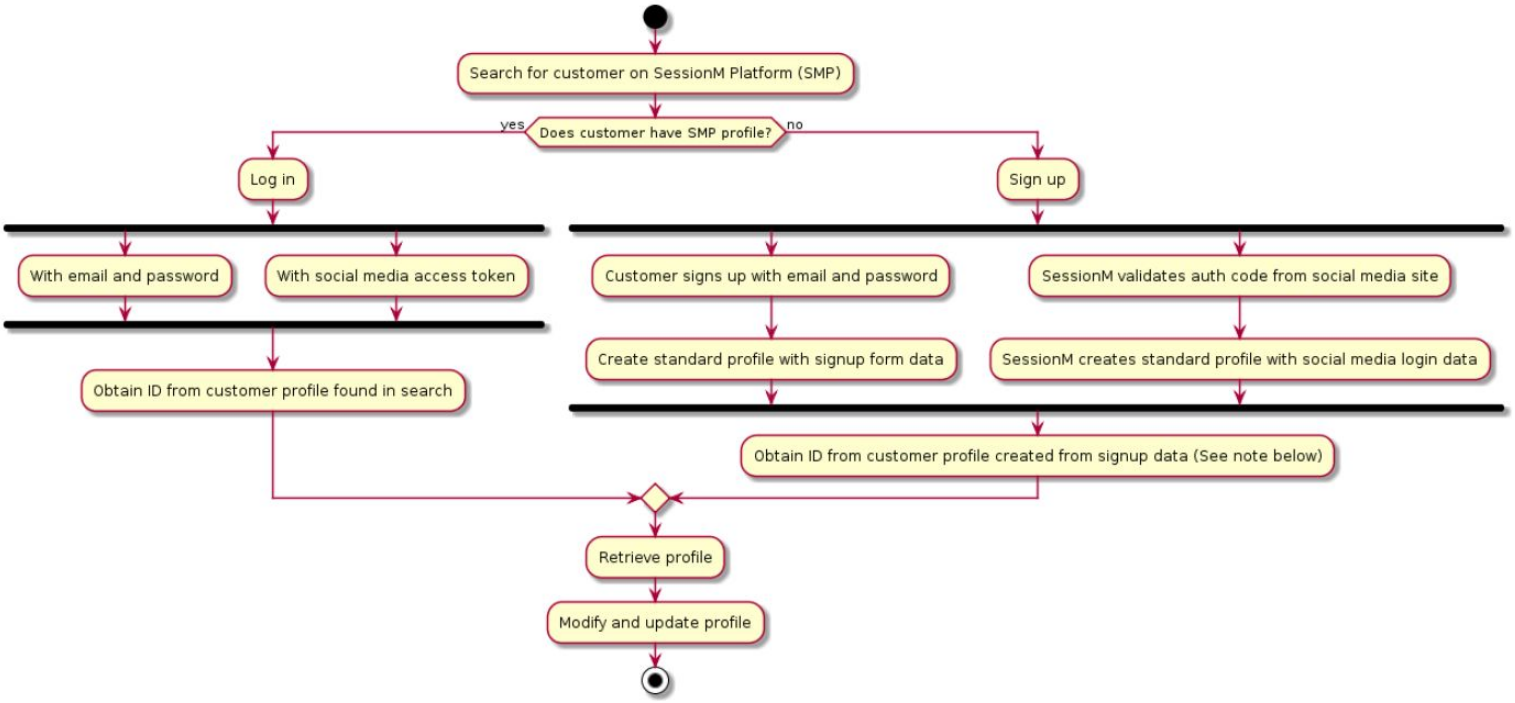
Data Configuration Prerequisites

This workflow presumes the client has performed the following tasks before it can be implemented for both sign-ups and logins:

- Creates and hosts a signup form that customers can access.
    - If signups are through a social media site, the signup form must be integrated with the social login module. This can be done by using the social provider's web SDK.
    - Be sure to update signup form/site when a new social provider is being used, and share the social media site's application ID and credentials with SessionM.
- Sets up server to implement the SessionM S2S APIs.

Step by Step

The workflow for signing up or logging in a customer follows the step-by-step decision tree diagrammed below. The top part of the workflow diagram details the steps associated with logging in or signing up a customer in an S2S implementation:



Note: After a customer has signed in, they can opt to log in as captured in the "Log in" step shown in the top diagram.

The sections that follow reflect the decisions and actions detailed in this diagram. Note that throughout this workflow external IDs are featured in the specification of endpoints, but internal IDs can also be used.

When issuing curl commands for platform transactions, adhere to the following syntax:

- Begin each curl command with either `POST` or `GET`.
- Specify: `-H 'Content-Type: application/json' -H 'authorization: Basic AUTH_ID'`
- Begin URL with same endpoint + API key: `https://[ENDPOINT]/priv/v1/apps/API_KEY`

Step 1: Determine if the Customer Already Has a SessionM Profile

This workflow begins with determining whether your customer already has a profile on the SessionM Platform.

You can use the Standard Profile API to search for the customer by one of the typical attributes of a customer profile, such as date of birth, first name/last name combination, or an email address, which is shown below:

```
GET /priv/v1/apps/:api_key/users/search?email=test@example.com
```

After the endpoint makes the request for the query, the platform returns a response object with the profile, which is shown below:

Response


```
{
      "status": "ok",
      "user": {
            "id": "e76fc440-d7f0-11e8-91e4-469b02598280",
            "external_id": "88888888",
            "opted_in": true,
            "activated": false,
            "proxy_ids": [],
            "email": "zzz@example.com",
            "gender": "m",
            "dob": "1980-01-01",
            "created_at": "2018-10-25 00:57:11",
            "updated_at": "2018-10-25 00:57:11",
            "address": "7 Tremont Street",
            "city": "Boston",
            "zip": "02021",
            "dma": "506",
            "state": "MA",
            "country": "USA",
            "suspended": false,
            "last_name": "Smith",
            "first_name": "ZZ",
            "registered_at": "2018-10-25 00:57:10",
            "profile_photo_url": "/images/account-neutral.png",
            "test_account": false,
            "account_status": "good",
            "tier_levels": [],
            "referrer_code": "ZZS-B27C94",
            "phone_numbers": [{
                  "phone_number": "1234123123",
                  "phone_type": "home",
                  "preference_flags": ["primary"],
                  "verified_ownership": false
            }]
      }
}
```


If they have a profile, they can log in to the client site two different ways:

- Directly, by submitting their email and password as login credentials.
- Indirectly, by submitting their social media site access token.

For this flow, go to Step 3.

If they do not have a profile, they can sign up two different ways:

- Directly, by submitting email and password via a signup form.
- Indirectly, by logging in to a social media site.

For this flow, go to Step 2.

Step 2: Create a Customer Profile with Signup Data

If you are following the flow that supports a customer signing up for a customer profile, you can build the signup logic to support two different scenarios.

Creating a Profile with Data from a Social Media Site

You can have SessionM create a customer profile using social media login data. For more information, contact your integration engineer.

Creating a Profile with Data from a Client's Signup Form

You can create a customer profile using signup form data that the customer provided after being authenticated by their email and password. In this step of the workflow a customer accesses a client's UI via a signup button sent to them in an email. The data the customer provides on the signup form becomes a standard customer profile.

Using the Standard Profile API, specify this endpoint to create a new standard profile with data from the signup form:

```
POST /priv/v1/apps/:api_key/users
```

The endpoint passes in the *user* request object:

Request

```
{
    "user":{
        "external_id":"88888888",
        "opted_in":"true",
        "external_id_type":"facebook",
        "email":"zzz@example.com",
        "first_name":"John",
        "last_name":"Smith",
        "password":"Aaaaaaaa1",
        "Gender":"m",
        "dob":"1980-01-01",
```

```
        "address":"7 Tremont Street",
        "city":"Boston",
        "state":"MA",
        "Zip":"02021",
        "country":"USA",
        "phone_numbers":[{
            "phone_number":"1234123123",
            "phone_type":"home",
            "preference_flags":["primary"]}]}
}
```

Notice the key attributes in the request (derived from the signup form), including *email* and *password*.

The platform returns a response object that contains the newly created profile:

Response


```
{
      "status": "ok",
      "user": {
            "id": "e76fc440-d7f0-11e8-91e4-469b02598280",
            "external_id": "88888888",
            "opted_in": true,
            "activated": false,
            "proxy_ids": [],
            "identifiers": [{
                  "external_id": "88888888",
                  "external_id_type": "facebook"
            }],
            "email": "zzz@example.com",
            "gender": "m",
            "dob": "1980-01-01",
            "created_at": "2018-10-25 00:57:11",
            "updated_at": "2018-10-25 00:57:11",
            "address": "7 Tremont Street",
            "city": "Boston",
            "zip": "02021",
            "dma": "506",
            "state": "MA",
            "country": "USA",
            "suspended": false,
            "last_name": "Smith",
            "first_name": "John",
            "registered_at": "2018-10-25 00:57:10",
            "profile_photo_url": "/images/account-neutral.png",
            "test_account": false,
            "account_status": "good",
            "tier_levels": [],
            "referrer_code": "JOHNS-708C6E",
            "phone_numbers": [{
                  "phone_number": "1234123123",
                  "phone_type": "home",
                  "preference_flags": ["primary"],
                  "verified_ownership": false
            }]
```

```
        }
}
```

In addition to several personal attributes about the customer, this response includes a generated ID. In the sample above, that ID is *e76fc440-d7f0-11e8-91e4-469b02598280*. This ID can be utilized to generate an access token when you log onto the platform in the next step or when you retrieve one in Step 4.

Step 3: Log in to the Platform

Often a customer needs to login to the platform. Maybe too much time has elapsed since their initial signup; maybe they want to login from a different device or web site. Either way, the access token needs to be re-issued so the customer accessing the platform from a client app or web site needs to be invited to login again.

Once the new access token has been acquired, you can facilitate whatever transactions are required on the customer's behalf. If you are following the flow that supports a customer signing up for a customer profile, you can build the signup logic to support two different scenarios.

Logging in with Social Media Credentials

You can log in using the credentials authenticated via a social media site as well as the associated access token. For more information, contact your integration engineer.

Logging in with Email and Password

You can log in using the email and password credentials the customer provided after being authenticated on the client's site.

Using the Identity Services API, specify the following endpoint:

```
POST /<identity service URL>/oauth/token
```

The endpoint passes in this request object:

Request

```
{
    "grant_type":"password",
    "email":"zzz@example.com",
    "password":"Aaaaaaaa1",
    "client_id":"c5c544350993cdb685649da2b42f0e407f8689c13391a3d681e5bb00e",
    "scope":"openid profile email"
}
```

Note that the *grant_type* attribute is set to "password" and values for *email* and *password* are provided.

If you want to simply refresh the token you have already, you can use the same endpoint with slightly different parameters defined in either a request object or as endpoint parameters. For more information, see this method in the Identity Services API.

After the endpoint makes the request for the, the platform returns a response object, which is shown below:

Response


```
{
      "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiIyMDE4LTEwLTI1IDAxOjEzOjE4ICswMDAwIiwiZXhwIjoiM
jAxOC0xMS0wOCAwMToxMzoxOCArMDAwMCIsImRhdGEiOnsiaWQiOiIyODE0YTA5MC1kN2YzLTExZTgtODkzNy1lMDMwMjU1
MWYyMTgifX0.seP9hmQ1xoD6M3NT0XN58iX8YtKLqKDjr2CRkdRUAFRf3KLYvJEuGDAJmSRFtHZo2fZwt3KAnRfpfXMuWA-
o2Pwq3UobfBerFvH512vzyySiDK9bM8kpzJGeV3azJZS1fjNVd2Ic5Y7QhvIgTuzyJMk8_0jFc9L9b1PqoABhQbIW96lpmw
jF1tEl2mjYoUamtY-wvzRHlQgdUvVN3BPTN8Z1IY8mYbvy1ndIveSjDm-5rm9DX-qU5qYK0ZwqhB5RuY58_JnAe3uAdnbeS
mZ5qLy89GhuAp3cwoV6fg4Y_ggpmDJlnNVbSNY91erlsxGv5gam4qLzn-v790T2IrbbuQ",
      "token_type": "bearer",
      "expires_in": 1209600,
      "refresh_token": "55f9a672f9107c4a3efb2b2b1a58b0a0c2ab6e1a3a28d661e",
      "scope": "openid profile email",
      "created_at": 1540429998,
      "id_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6InowMGdwUzZHUkxlV2M4ZExlUzFmRFZzRFpkNXhqbzltbHE5Z3
dKNmMwTWcifQ.eyJpc3MiOiJodHRwczovL2xvZ2luLWVjb25vbXkuc3RnLXNlc3Npb25tLmNvbSIsInN1YiI6ImU3NmZjND
QwLWQ3ZjAtMTFlOC05MWU0LTQ2OWIwMjU5ODI4MCIsImF1ZCI6ImM1YzU0NDM1MDk5M2NkYjY0NTA4NTY0OWRhMmI0MmYxM
mNmMGU0MDdmODY4OWMxMzM5MWEzZDY4MWU1YmIwMGUiLCJleHAiOjE1NDA0MzAxMTgsImlhdCI6MTU0MDQyOTk5OCwiYXV0
aF90aW1lIjoxNTQwNDI5OTk4fQ.B5iBJTcThDkNacYgwYHmo3e_g1_tJILY7O6mKi40qkoTbnKL6_UKwDMpHJ8_11ihbNW9
UC3O7l9ut-H_VWJHodXQM6hbvaZqBs4GHxbYwQyqkjge4i3XNGovc_VCSUrfia9WWBzuvA-mpXeh9-6e2yc16g80R4aW1Cr
_0Mw2_eOemFQebO12S4J31lIjKfwTXFuHcOmYMZ2S-KyX63NRrh_XcZM0S7uETX2oG_cMXfVjQ2SseSXBau3KuBEH1EzPXy
36rUXIkv5qrBRkYXrdkVWH2-H5PTlsXfiChY6WviLB0fZdAV3HvK8SnlaADmCwDtuGqM7bvU8UJdWqXhsDtg"
}
```


Note that the response contains the access token, which you can use to retrieve a customer profile in Step 5.

Step 4: Obtain ID for S2S Transactions

Once a customer has a customer profile, that profile can be updated. The next step is to acquire the ID associated with the customer so you can then retrieve the profile.

Server-to-Server transactions reflect implementations that feature a client server initiating communication with the SessionM Platform. To perform any of these transactions, such as updating a profile, you need to get the ID associated with the customer profile.

If logged in, the customer's ID can be obtained from the response produced by having performed a search, which is shown in Step 1. So for logged in customers, simply pull the ID from the search response.

Alternatively, you can also get the ID that's provided in the response to creating a profile from the personal data the customer provided when they signed up. For more information, see Creating a Profile with Data from a Client's Signup Form.

With this ID, you can perform a variety of transactions, may of which require the retrieval of a customer's profile. For more information, proceed to Step 5.

Step 5: Retrieve a Customer Profile

The next step in the workflow is retrieving the customer profile so you can update it. Using the Standard Profile API, specify the following endpoint:

```
GET
/priv/v1/apps/33580ff1eedb5a8b7c9c23c8db4e766e707350e4/users/e76fc440-d7f0-11e8-91e4-469b02598280
```

Note that this endpoint identifies the customer by the same ID used to retrieve an access token; in this workflow, *e76fc440-d7f0-11e8-91e4-469b02598280*. Be sure to put the access token in the request header.

After the endpoint makes the request for the customer profile, the platform returns a response object, which is shown below:

Response


```
{
      "status": "ok",
      "user": {
            "id": "e76fc440-d7f0-11e8-91e4-469b02598280",
            "external_id": "88888888",
            "opted_in": true,
            "activated": false,
            "proxy_ids": [],
            "email": "zzz@example.com",
            "gender": "m",
            "dob": "1980-01-01",
            "created_at": "2018-10-25 00:57:11",
            "updated_at": "2018-10-25 00:57:11",
            "address": "7 Tremont Street",
            "city": "Boston",
            "zip": "02021",
            "dma": "506",
            "state": "MA",
            "country": "USA",
            "suspended": false,
            "last_name": "Smith",
            "first_name": "John",
            "registered_at": "2018-10-25 00:57:10",
            "profile_photo_url": "/images/account-neutral.png",
            "test_account": false,
            "account_status": "good",
            "tier_levels": [],
            "referrer_code": "JOHNS-708C6E",
            "phone_numbers": [{
                  "phone_number": "1234123123",
                  "phone_type": "home",
                  "preference_flags": ["primary"],
```

```
                "verified_ownership": false
            }]
      }
}
```

With the profile retrieved, it can be updated. Notice that in this response the first name of the customer is “John.” In the next step, you can modify this attribute.

Step 6: Modify and Update a Customer Profile

The next step in the workflow is modifying and updating the customer profile. Using the Standard Profile API, specify the following endpoint:

```
PUT
/priv/v1/apps/33580ff1eedb5a8b7c9c23c8db4e766e707350e4/users/e76fc440-d7f0-11e8-91e4-469b02598280'-d'{"user":{"first_name":"ZZ"}}
```

Note that this endpoint replaces the value “John” with “ZZ” for th *first_name* attribute.

After the endpoint makes the request for the customer profile update, the platform returns a response object, which is shown below:

Response


```
{
      "status": "ok",
      "user": {
            "id": "e76fc440-d7f0-11e8-91e4-469b02598280",
            "external_id": "88888888",
            "opted_in": true,
            "activated": false,
            "proxy_ids": [],
            "email": "zzz@example.com",
            "gender": "m",
            "dob": "1980-01-01",
            "created_at": "2018-10-25 00:57:11",
            "updated_at": "2018-10-25 00:57:11",
            "address": "7 Tremont Street",
            "city": "Boston",
            "zip": "02021",
            "dma": "506",
            "state": "MA",
            "country": "USA",
            "suspended": false,
            "last_name": "Smith",
            "first_name": "ZZ",
            "registered_at": "2018-10-25 00:57:10",
            "profile_photo_url": "/images/account-neutral.png",
            "test_account": false,
            "account_status": "good",
            "tier_levels": [],
```

```
            "referrer_code": "ZZS-B27C94",
            "phone_numbers": [{
                  "phone_number": "1234123123",
                  "phone_type": "home",
                  "preference_flags": ["primary"],
                  "verified_ownership": false
            }]
      }
}
```

The profile has been updated with the customer's new first name.