SAP Customer Data Cloud

ILT Exercises

TABLE OF CONTENTS

Welcome: SAP Customer Data Cloud Instructor-Led Training Exercises

Introduction This course will provide you with a fundamental knowledge of SAP Customer Data Cloud and its functionalities. Through hand-on experiences, you'll learn how to perform common implementation-related tasks.

Objectives Upon completion of the course, you'll be able to:

- Recognize the key features of the SAP Customer Data Console to manage the site configuration and access tools that display, query, and analyze your user database.
- Configure and design data store schema to easily search for any stored fields in the data.
- To implement Lite Registration flow and access Lite Registration data through Identity Access.
- To create full accounts by following various implementation approaches.
- Discover how to utilize REST APIs to consume Customer Data Cloud services and implement various flows.
- Discover the various criteria and actions that can be applied to login attempts using RBA rules to add an extra layer of account security to your site.
- Implement SAP Customer Consent
- Define JavaScript Parameters to give you greater flexibility when customizing the user interaction with your SAP Customer Data Cloud screens.
- Set up Dataflows to sync consent-based user data to third party applications using the exercise steps.
- Recognize the formats used to securely transmit information between parties as a JSON object.

Exercise Preparation

Setting up your system

In this document, we are going to explain how to set up your system to perform the exercises.

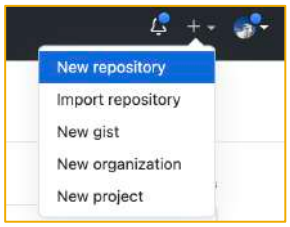
TASK 1: Set up your system.

Preparation

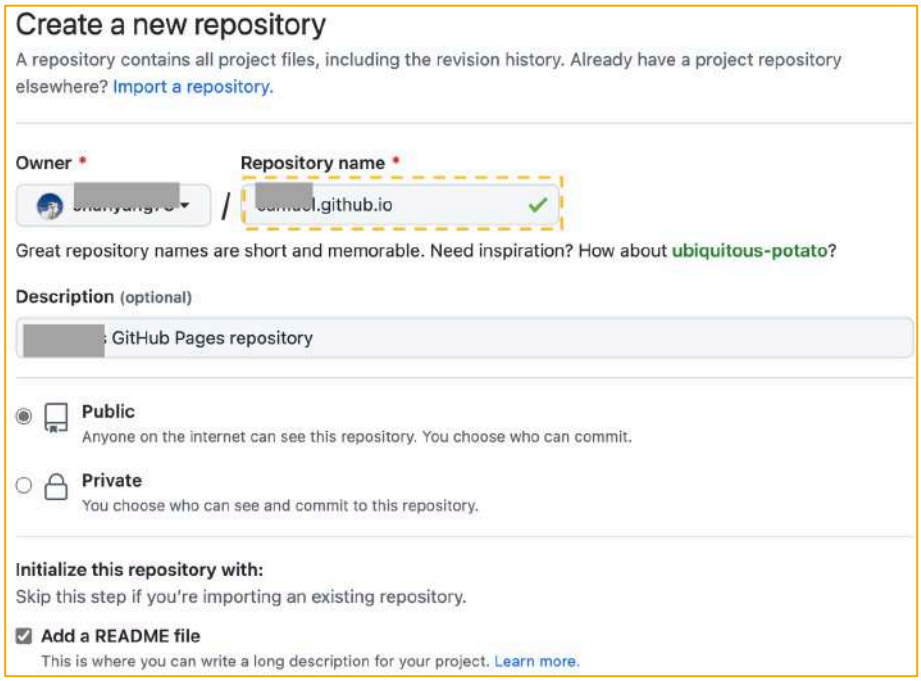
- You may already have a GitHub account and a public repository. If you don't have a GitHub account, please go to https://github.com/ and create a new account.
- Follow these, if:
    - You're not sure where that directory is located,
    - You don't have a web server running on your computer,
    - You're not sure how to create a repository and a GitHub Pages site:

To create a GitHub repository, follow these steps:

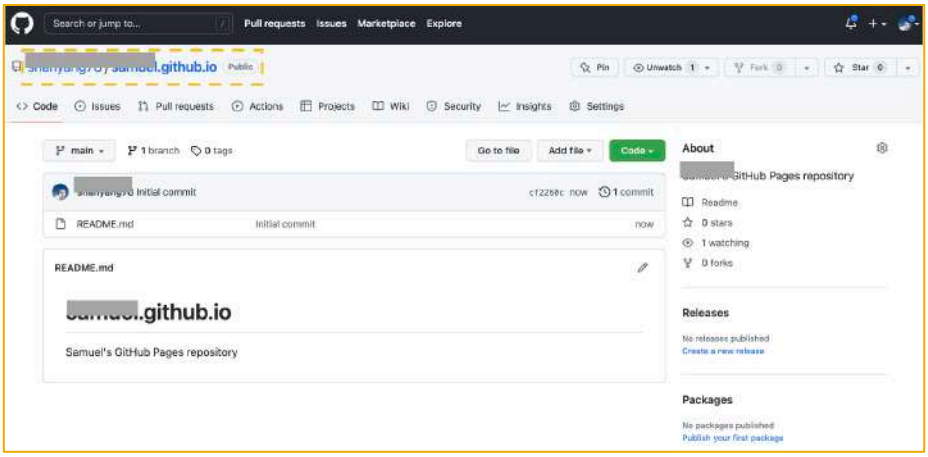
1. Login to **GitHub**.
2. Click the + dropdown button on the top right corner.
3. Select **New Repository**.



4. On the Create a new repository page
    - Give your repository a name like **YOUR_USERNAME.github.io**.
    - Replace YOUR_USERNAME with **your name**.
    - Keep the **Public** repository option.
    - Optionally select **Add a README file**.



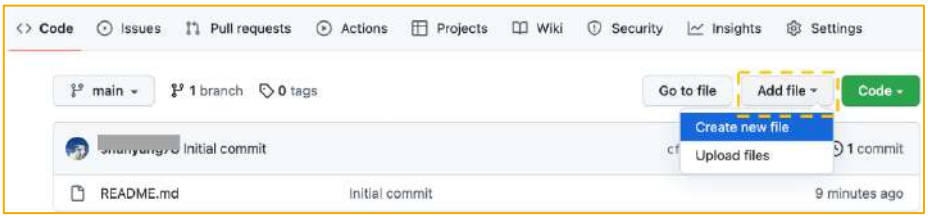
5. Click **Create repository** at the bottom of the page, a new repository is created. By default, the GitHub Pages feature is not enabled for the new repository.



Task 2: Create a GitHub Pages site

To create a GitHub page site, follow these steps:

1. On the new repository page, click on the **Add file** dropdown.
2. Select **Create new file**.

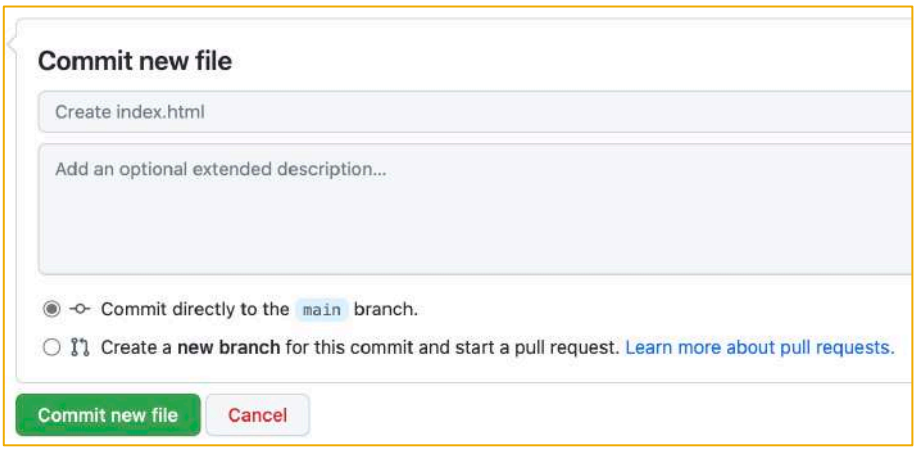


3. On the Edit new file page**,** copy & paste the following HTML codes in the code editing area.

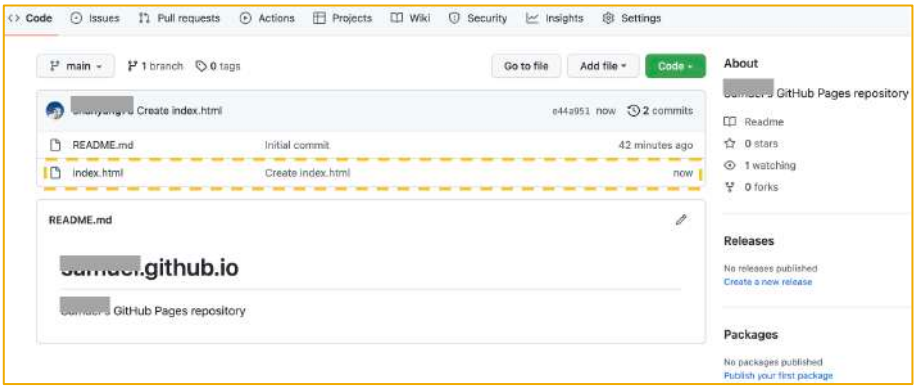
```
<!DOCTYPE html>

<html>

  <head>

    <title>My GitHub Pages Site</title>

  </head>

  <body>

    <h1>Hello SAP Customer Data Cloud!</h1>

    <p>Welcome to my GitHub Pages site!</p>

  </body>

</html>
```

4. Enter **index.html** as the file name next to the repository name xxx.github.io/.

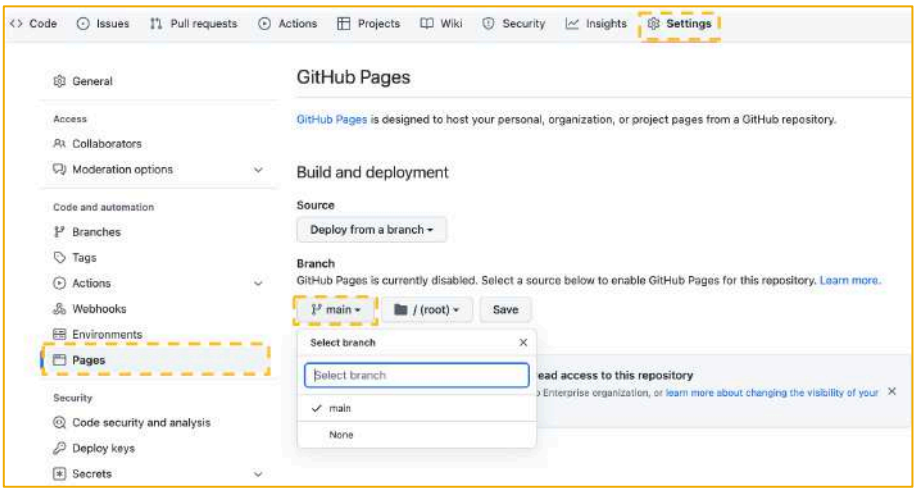
5. Scroll down to the bottom of the page and click **Commit new file**.



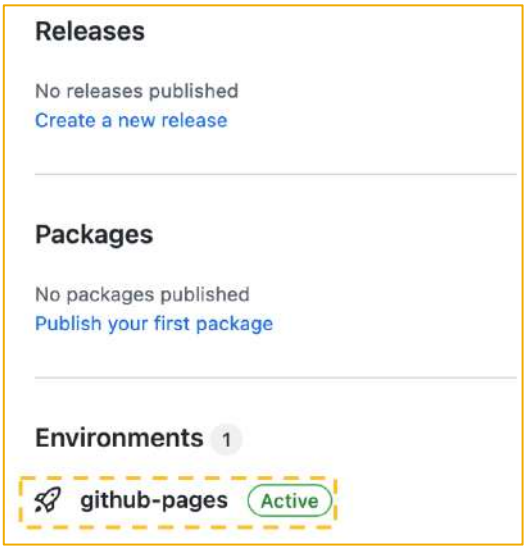
The index.html file is created in the repository.



6. GitHub Pages
   - Click on the **Settings** button.
   - Select **the Pages** tab from the left side.
   - Keep **Deploy from a branch** as the default publishing source.
   - Click the **Branch** dropdown and select **main**.
   - Keep the folder as **/(root)**. For the sake of simplicity, we're not using child branches or subfolders to host the HTML files.
7. Click **Save**.

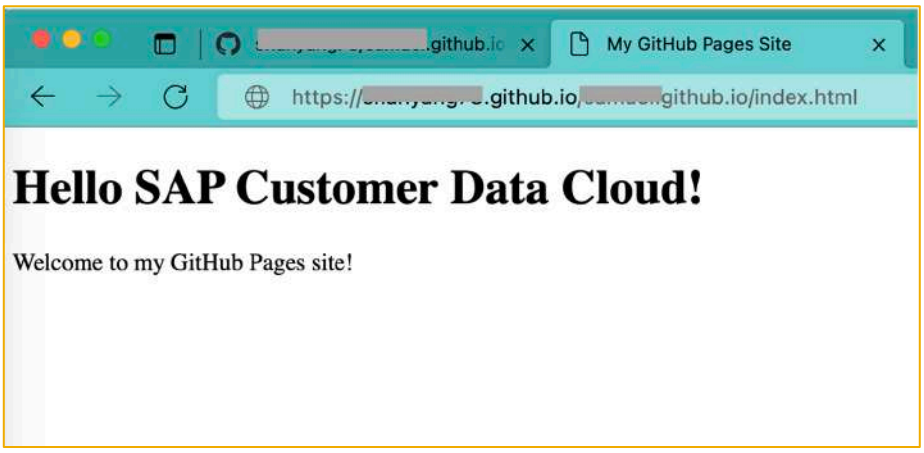


Back on the repository page, you will see the GitHub Pages feature is now enabled.



8. Navigate your browser to **https://your_gh_account.github.io/your_name.github.io/index.html** and your HTML page is served right from GitHub.

Now you're ready to work on the exercises.



And you are ready to dive into CDC world and have some hands-on in the console.

Exercise 1: Customer Data Cloud Overview

Activate Customer Data Cloud Account

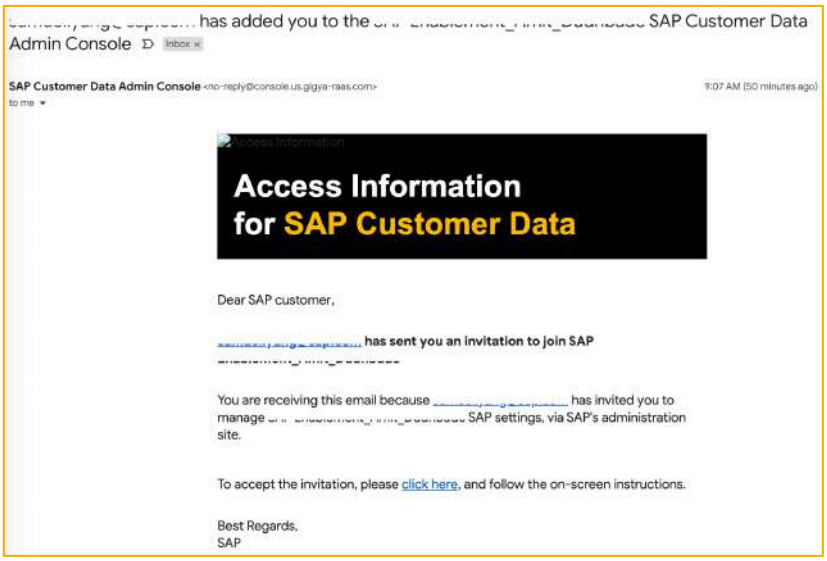
In this exercise, you'll activate your account and access the customer Data Cloud console. Accept the invitation and log into the Customer Data Cloud Console.

Task 1: Activate Customer Data Cloud Account

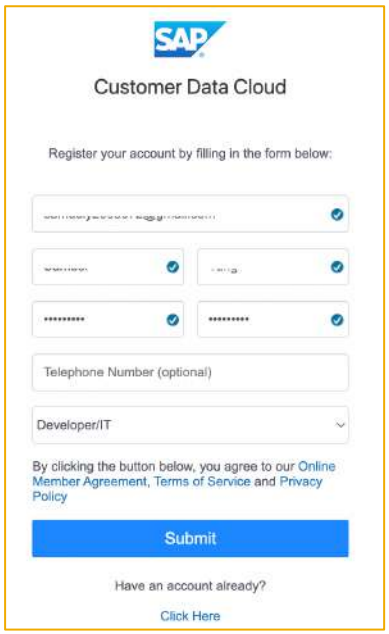
Solution

To activate the Customer Data Cloud Account, follow these steps:

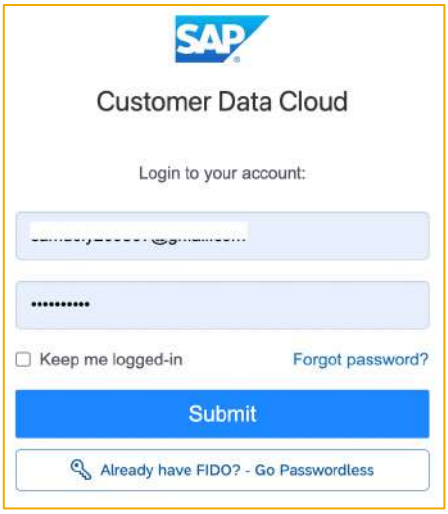
1. You will receive an email from Customer Data Cloud with a link to activate your account.



2. Click on the **click here** link
   - Complete the process.
   - Set the password.
   - Make sure to use the telephone number of a phone you have with you to receive the verification code.



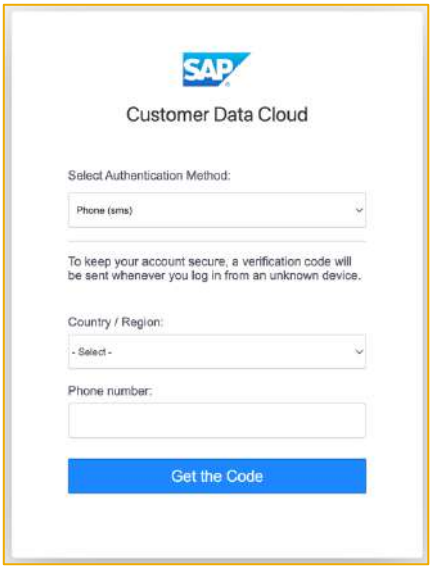
3. Make a note of the URL https://console.gigya.com/
4. Log into the **Customer Data Cloud Console**.



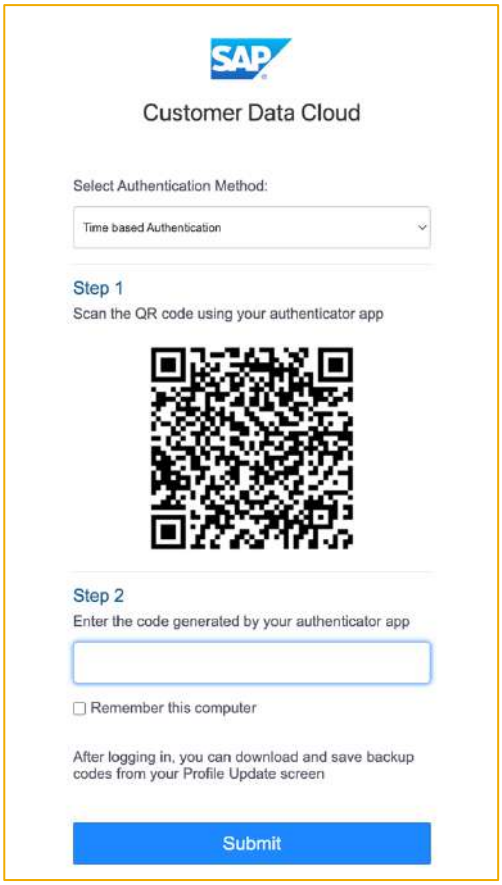
5. Two-factor authentication or TFA is enabled to access the Customer Data Cloud Console. In Select Authentication Method, you can pick Phone (sms) or Time based Authentication. If you decide to pick Time based Authentication, please install an authenticator app on your phone and follow the screen instructions.

   Enter **either Phone number or the authenticator app code**

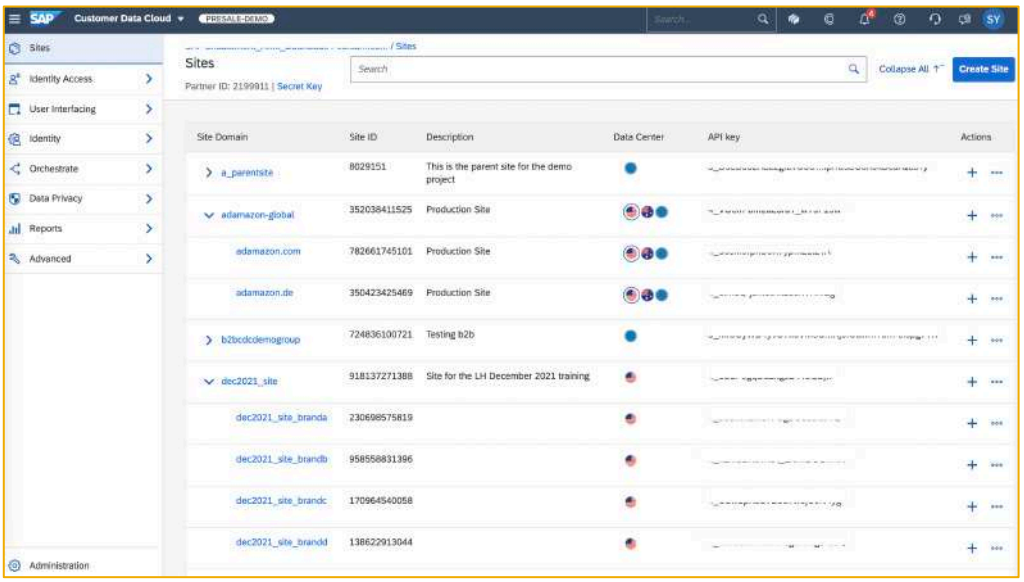
6. Click **Get the Code** and **Submit**.



Or



7. After successful login, you will get access to the Customer Data Cloud Console.



Recap

In this exercise, you have successfully logged into the Customer Data Cloud Console.

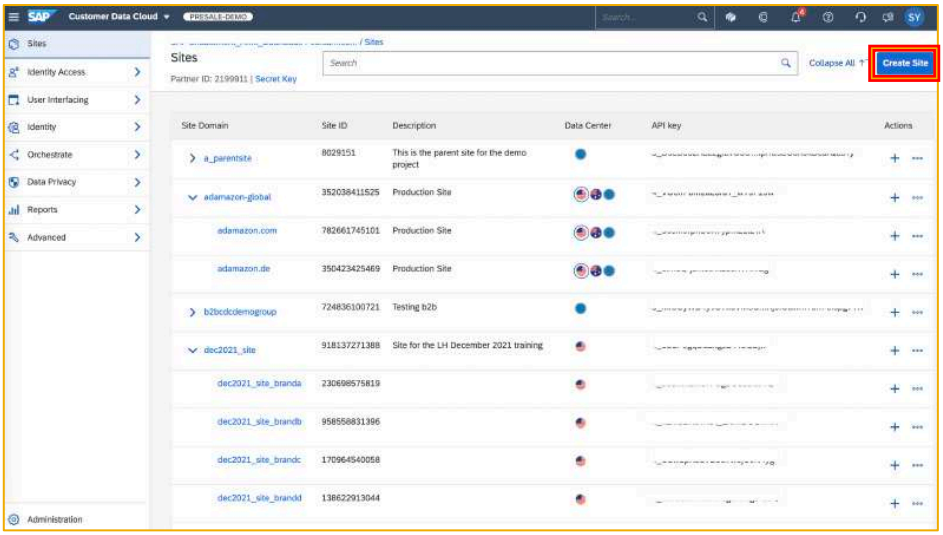
Task 2: Create your site.

Use SAP Customer Data Cloud to create your site for performing exercises.

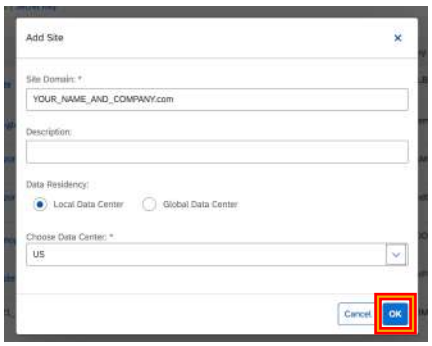
Solution

To create your site for performing exercises, follow these steps:

1. On the main screen of the SAP Customer Data Cloud console, click on **Create Site**.

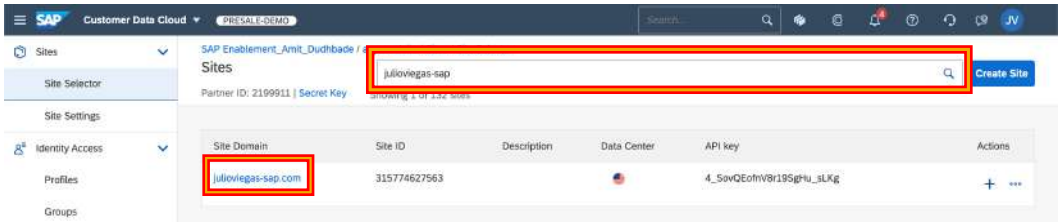


2. On the Add Site popup
   - Enter a **Site Domain**. You can use your own name & company name to fill in the placeholder YOUR_NAME_AND_COMPANY.
   - Data Residency: Select **Local Data Center**
   - Choose Data Center: Select the **Data Center closest to you** (Chinese residents must pick the CN Data Center).
   - Click **OK**.



Before every exercise, type in your site name to help find it faster.

3. In the results list, click on **your site** to go to its configurations and accounts information.



Recap

In this exercise, you have learned to access the Customer Data Cloud Console as a new user and how to create a new site.

Exercise 2: Data Schema

Access Account Schema

In this exercise, you will access the Accounts Schema to update profile fields and create new data and subscription fields in the Customer Data Cloud Accounts Schema.

Task 1: Edit Fields in schema

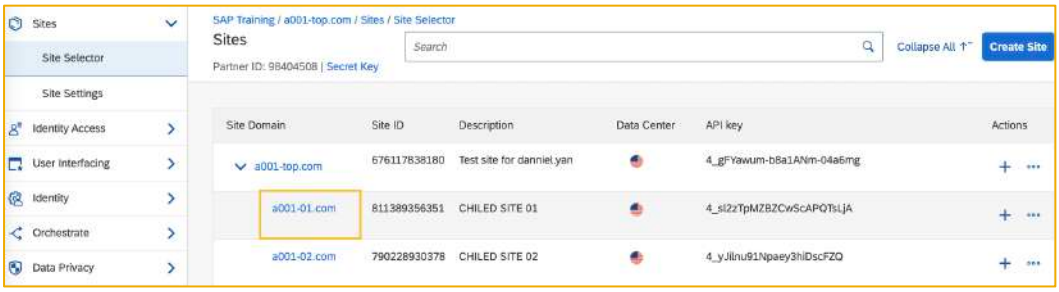
Identify the site assigned to you, open the Accounts Schema and make the following fields required.

- profile.firstName
- profile.lastName
- profile.email

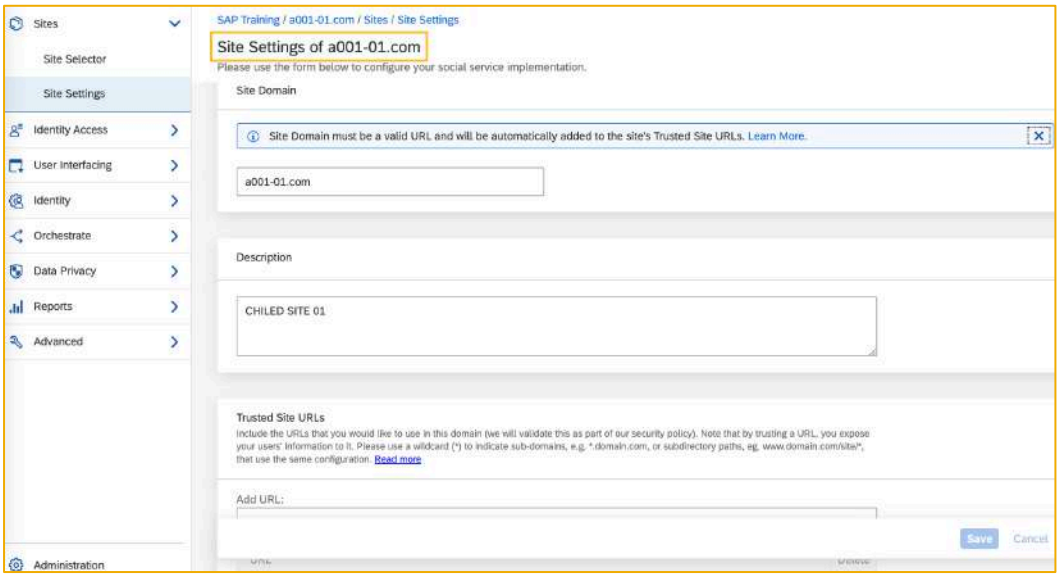
Solution

To edit fields in schema, follow these steps:

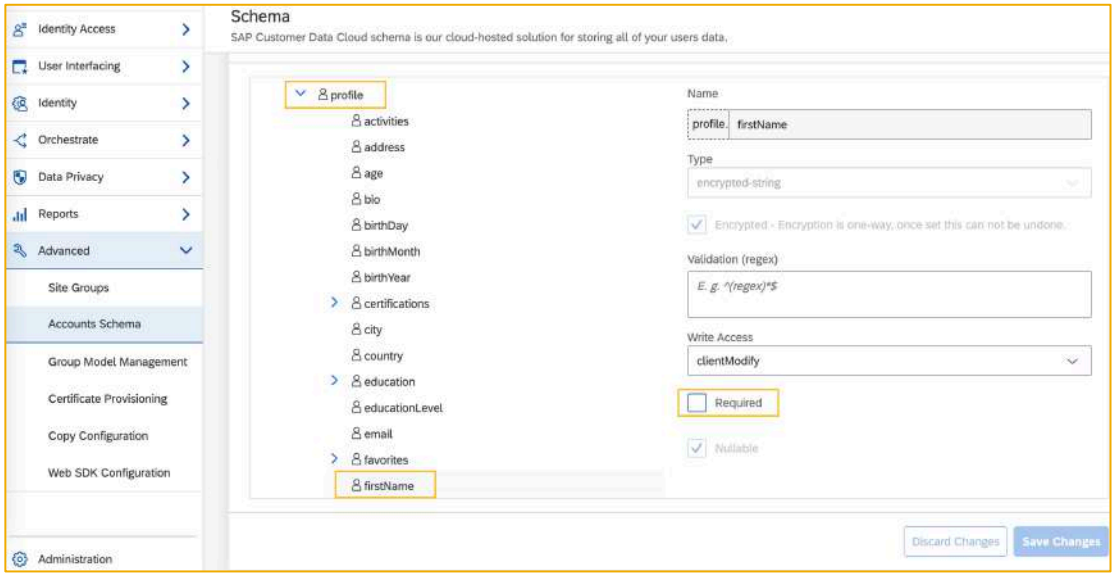
1. Access the Customer Data Cloud Console https://console.gigya.com and select Sites > Site Selector from the left side navigation menu.



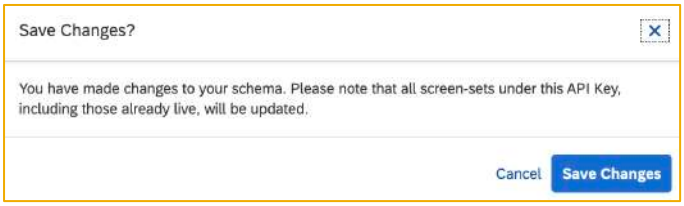
2. Select the site assigned to you in the site table. The Site Settings page opens up.



3. Navigate to the **Advanced** section on the left side navigation menu.
4. Click on the **Accounts Schema**. The Schema Editor opens up.
5. In the Schema Editor, look for **profile** under accounts database tree.
   - Select **firstName** under profile.
   - You will see field details on the right. Check the **Required** box.



6. Look for **lastName** and **email** fields and make them required as per step 5.
7. Click **Save Changes** at the bottom right of the page.
8. On the Save Changes? popup click on **Save Changes**.



Recap

In this exercise, you have learned to update profile fields!

Task 2: Create new fields in schema.

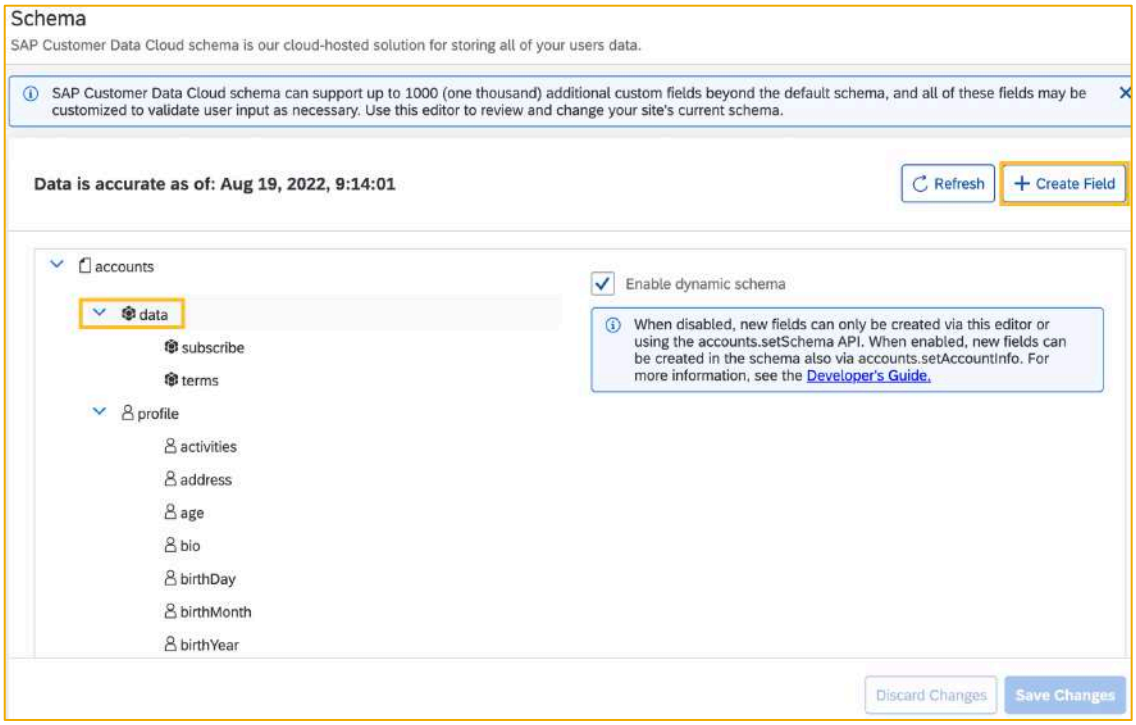
Create new data fields:

- data.title
- data.brand1.favSport

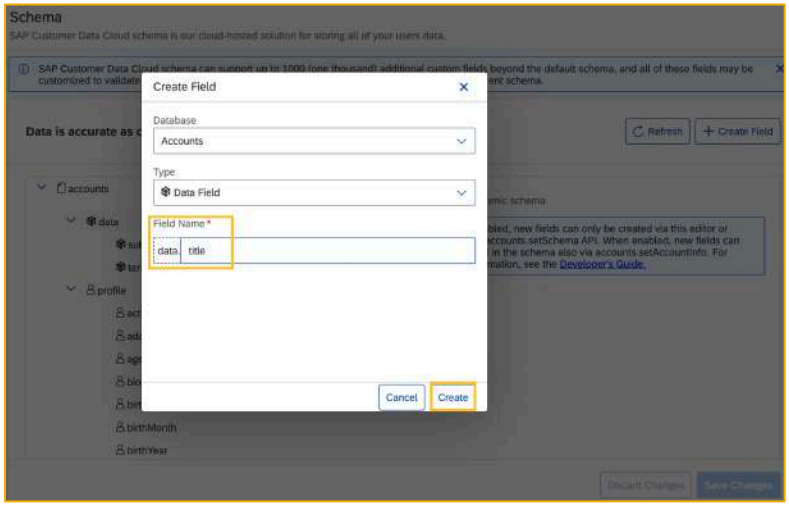
Solution

To create new data fields in schema, follow these steps:

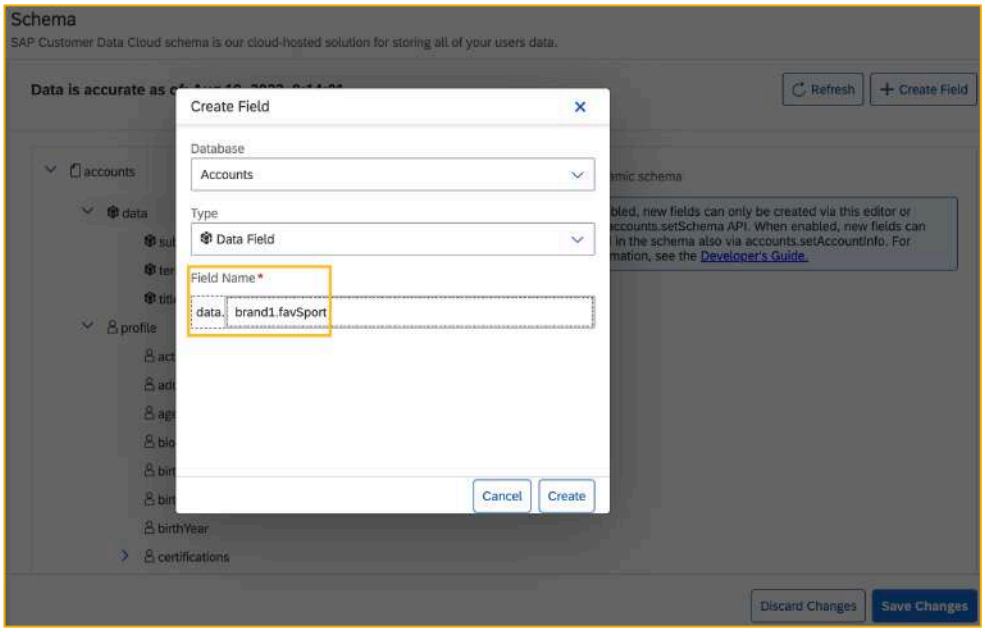
1. On the Schema Editor, select **data** object.
2. Click on the **+ Create Field** button.



2. Accounts database is selected. Field type is set to Data Field.
   - Enter the field name as **title**
   - Click **Create**.

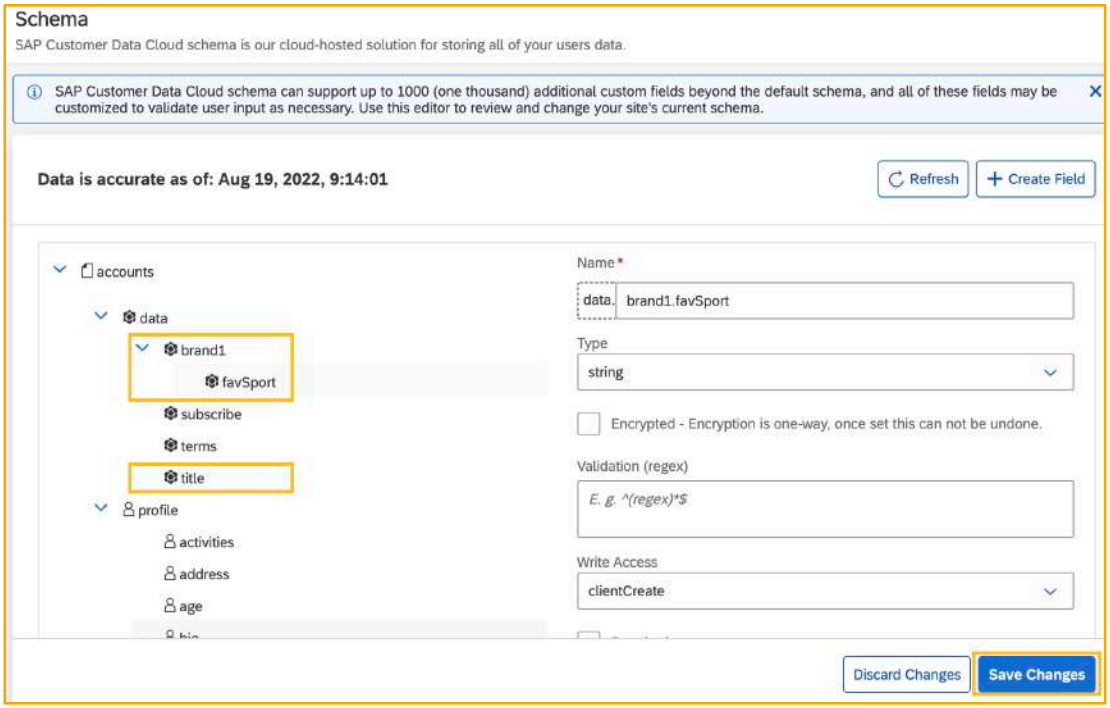


3. Create another field
   - Database: **Accounts.**
   - Type: Data Field.
   - Field Name: **brand1.favSport**.
   - Click **Create.**



Your schema should look similar to the one in the picture below.

4. Click on the **Save Changes**.



Recap

In this exercise you learned to create new data fields.

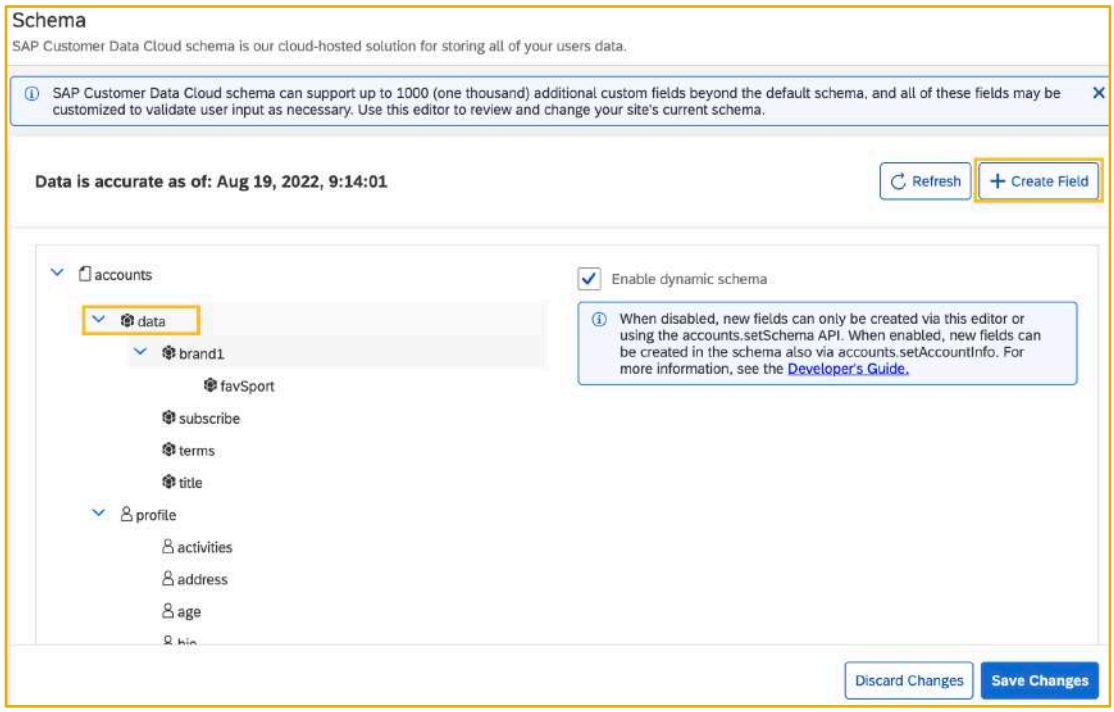
Task 3: Create subscription field.

Create a new subscription field WeeklyNewsletter and Enable the double-opt-In option.

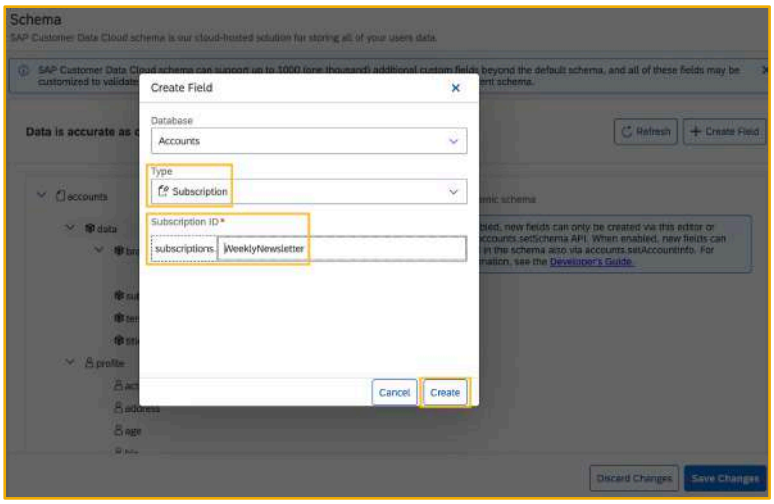
Solution

To create a subscription field, follow these steps:

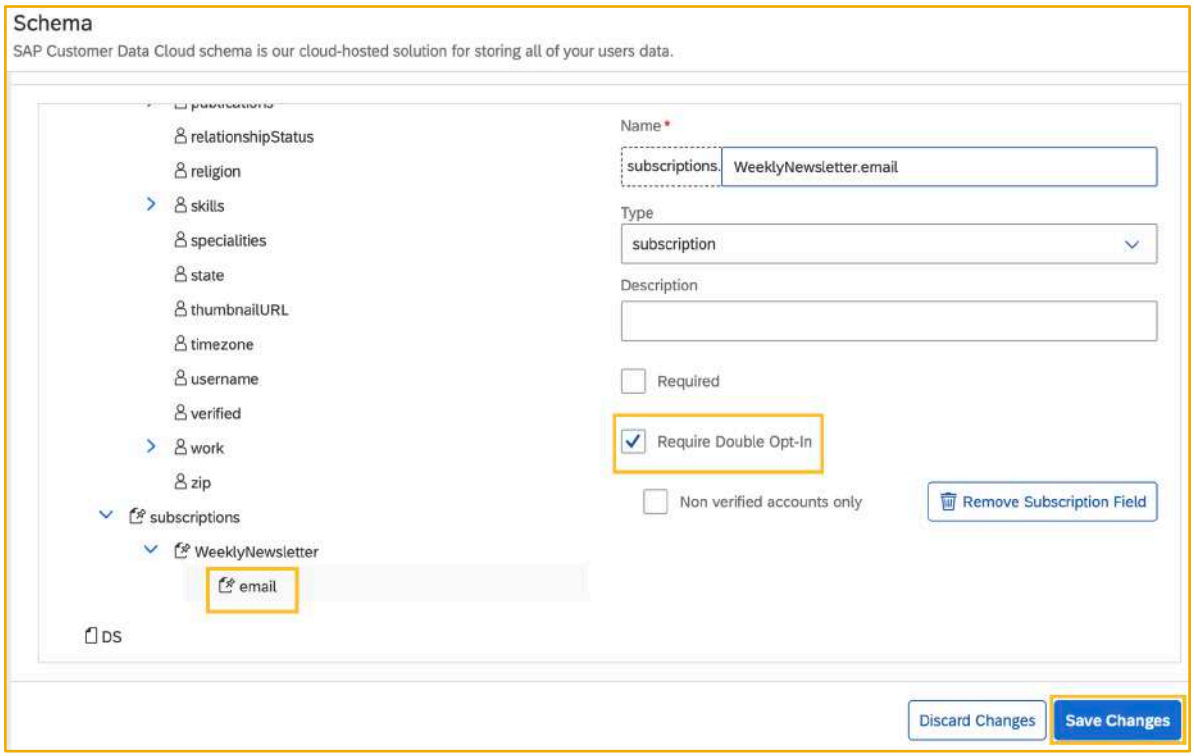
1. On the Schema Editor
   - Select the **data** object.
   - Click on **+ Create Field**.



2. Create Field
   - Keep the database as **Accounts**.
   - Change the Type to **Subscription**.
   - Enter **WeeklyNewsletter** in the Subscription ID field.
   - Click **Create**.



3. The WeeklyNewsLetter field is created under the Subscriptions object.
   - Select the **email** object.
   - Enable the **Require Double Opt-In** option for this field.
   - Click **Save Changes**.



Recap

In this exercise, you have learned how to change and extend the schema by setting the required fields, creating new data fields, and adding new subscription fields.

Exercise 3: Lite Registration

Enable Customer Identity Screen-sets

In this exercise, you will use and customize the out-of-box Customer Identity screen-sets to perform different types of customer registrations and logins. You will also learn how to localize the screen-sets and change the existing user flows using conditions and the Web SDK.

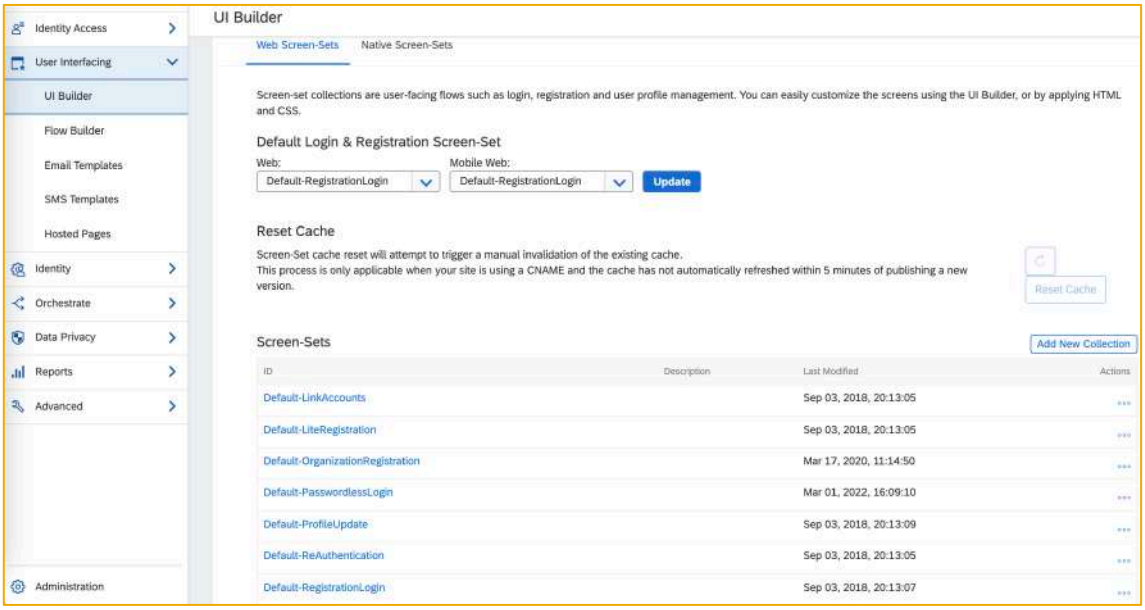
Task 1: Check Customer identity feature is enabled.

Open Site Settings – make sure the Customer Identity feature is enabled for you.

Solution

To check that the Customer Identity feature is enabled, follow these steps:

1. Select **User Interfacing** > **UI Builder** from the left side navigation menu to open UI Builder page.



The Customer Identity feature is enabled for your account if you see the screen sets under the Web Screen-Sets tab.

TASK 2: Lite Registration Flow.

In this exercise you will implement and customize the Lite Registration flow.

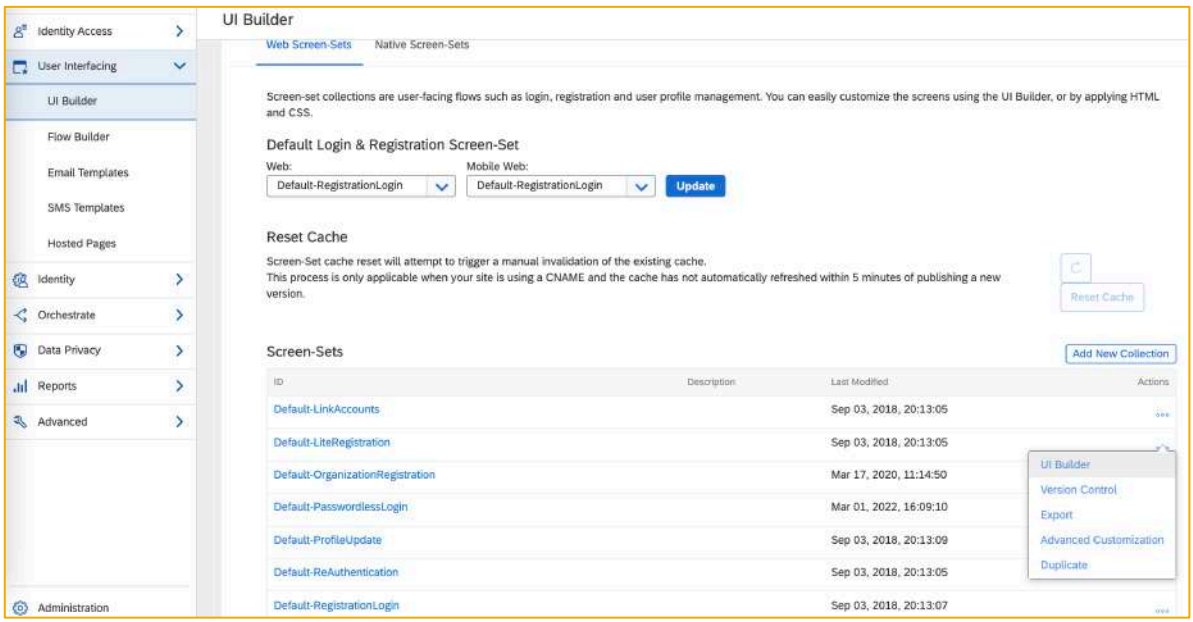
- Use the Default-LiteRegistration screen-set provided out of the box by SAP Customer Data Cloud.
- Bind the Weekly Newsletter checkbox available in the Default-LiteRegistration screen-set to a new Subscription field named WeeklyNewsletter created earlier.
- Create an index.html HTML file with the minimum JavaScript SDK blueprint code pointing to the API Key of our site. This page should contain a link that will trigger the Default-LiteRegistration screen-set flow when clicked.
- Then simulate a lite registration and verify the lite registered user information using the Identity Access tool available in the CDC Console.

Note: To test the html page, make sure you went through the *Exercise Preparation* document and performed all the steps.

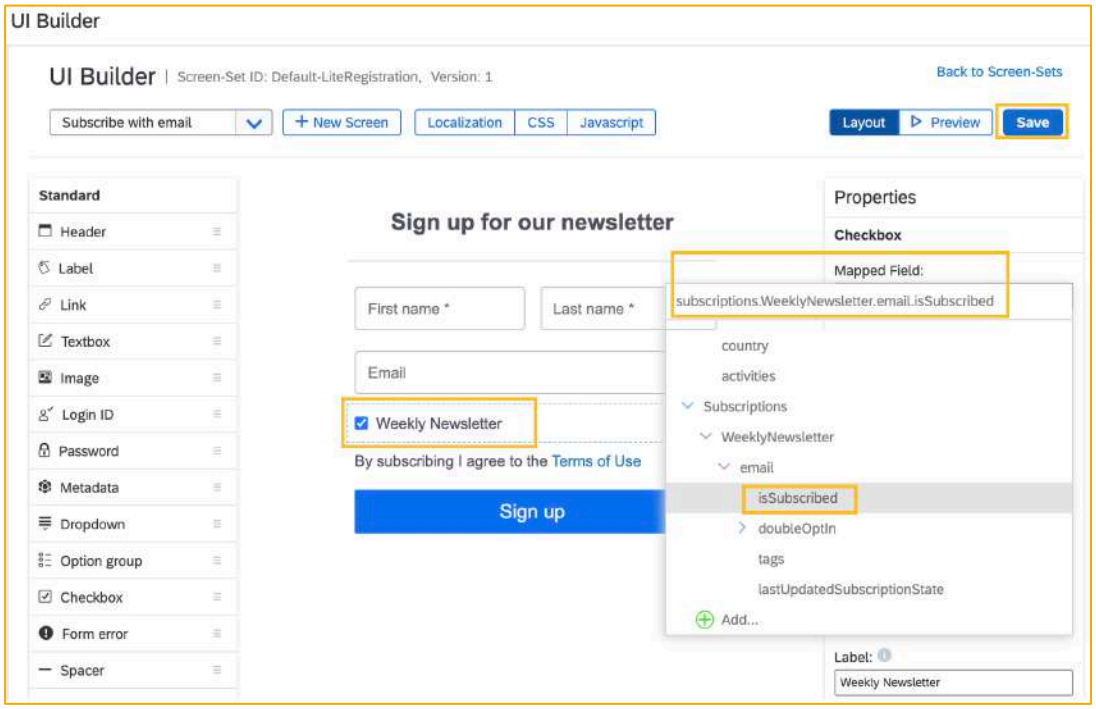
Solution

To implement and customize the Lite Registration flow, follow these steps:

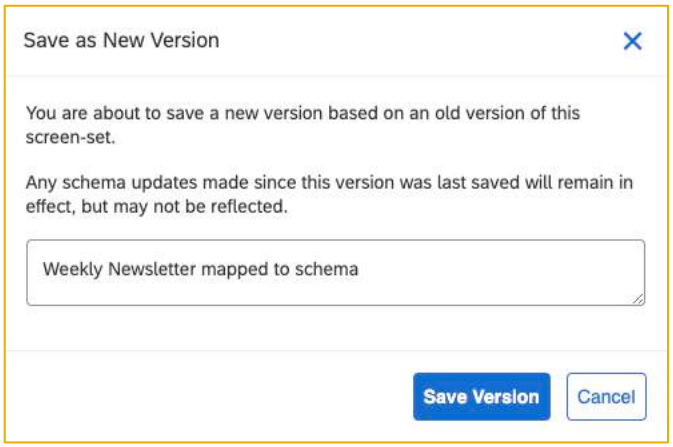
1. Access the Customer Data Cloud Console https://console.gigya.com
2. Select one of the sites assigned to you, e.g. cdc-student-1-2209.com
3. Select **User Interfacing** > **UI Builder** from the left side navigation menu.
4. On the UI Builder page, under the Web Screen-Sets tab, click on the **three-dot** action menu
   - Select **UI Builder** on the Default-LiteRegistration screen set.



5. Map the **Weekly Newsletter** checkbox to the **Weekly Newsletter** field in the Schema.



6. Click **Save**.
7. Enter a **description for your change**.
8. Click **Save Version**.

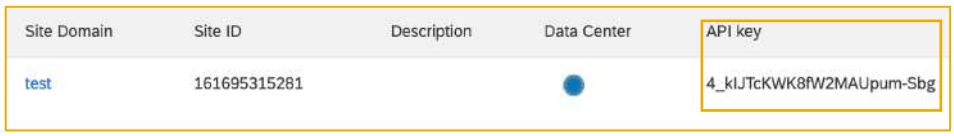


9. Click to open the **index.html file** at the root of your GitHub repository.
10. Click on the **pencil** button to edit the file.

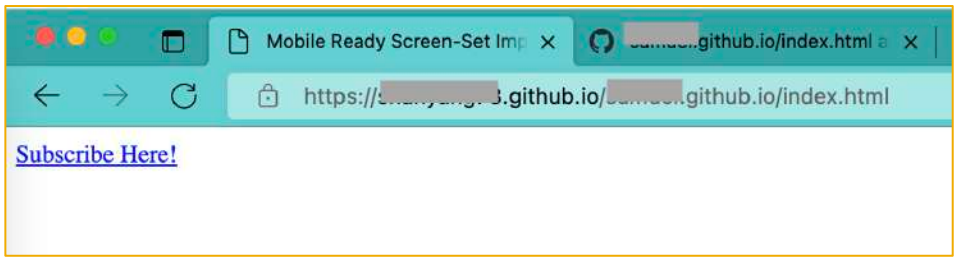
11. Copy & paste the **following HTML codes** to the **index.html** file. Make sure to **override everything** in the file if it's not empty.
12. Click **Commit changes** to save the file.

```
<html>

  <head>

    <title>Mobile Ready Screen-Set Implementation Example</title>

    <meta name="viewport" content="width=device-width">

    <SCRIPT type="text/javascript" lang="javascript"

            src="https://cdns.gigya.com/js/gigya.js?apikey=MY_API_KEY"></SCRIPT>

  </head>

  <body>

    <a href="#"

       onclick="gigya.accounts.showScreenSet({screenSet:'Default-
LiteRegistration'});">Subscribe Here!</a>

  </body>

</html>
```

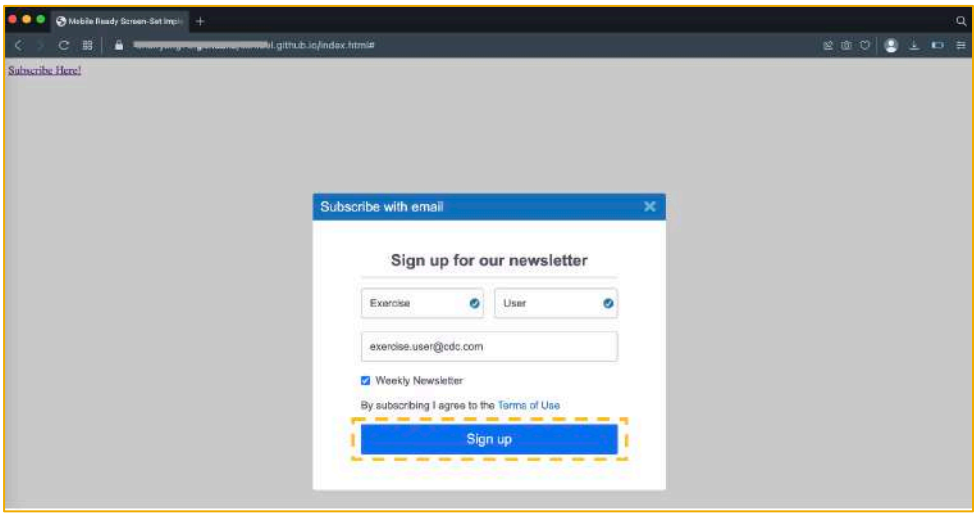
13. Go to the **Customer Data Cloud Console** home page.
14. Click over the **API Key** of the site you created to copy it to the clipboard.



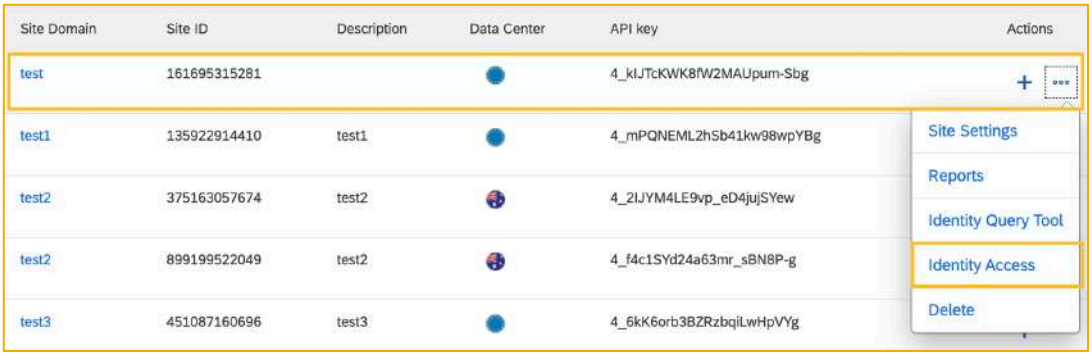
15. Go back to the **index.html** file.
16. Paste the **clipboard contents** to replace the string MY_API_KEY on line 6.
17. Save your file again.
18. Open the **HTML** file from GitHub in the browser.



19. Click on the **Subscribe here** link. The Subscribe with email pop-up appears. Enter the following:
    - First name
    - Last name
    - Email
    - Weekly Newsletter is already checked. Click on **Sign up**.
    - Click **OK** to dismiss the Thank you dialog.



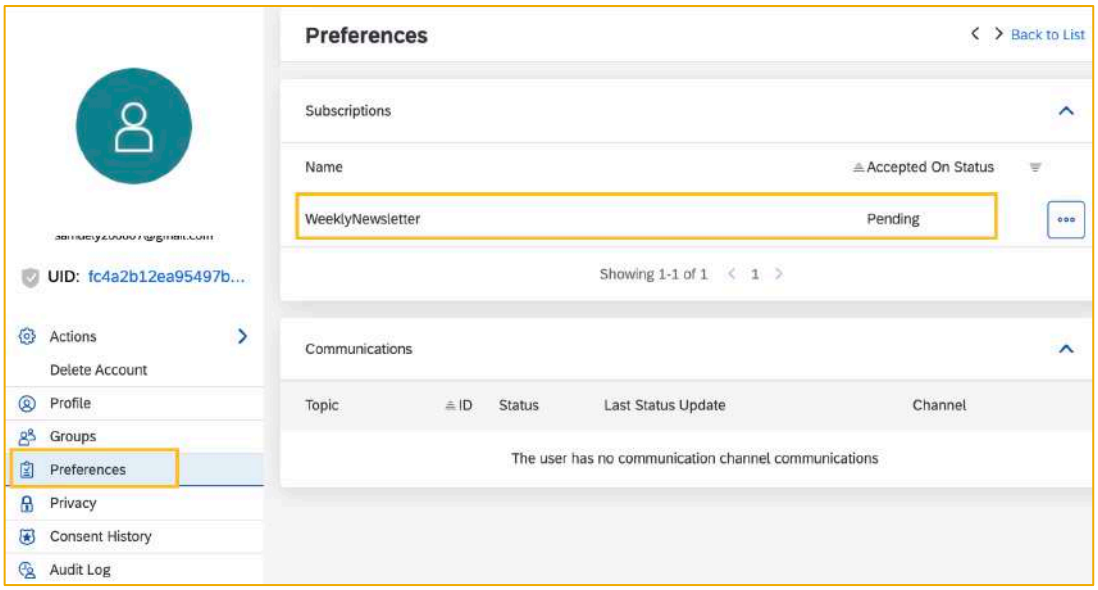
20. Return to the **Customer Data Cloud Console**, to the **Sites** table.
    - Select **Identity Access** located on the three dots actions menu at the right-hand side of your site.



21. The Identity Access page will show the user subscribed on step 16. Note the "**L**" symbol at the right of the user's row information. It signifies that the identity is a Lite registration.

22. Click on the **user** link.

23. The screen containing the user's account information will open. Click on the **Preferences** tab on the left side to verify if the user Opted in or out the Weekly Newsletter.



24. Once you accept the email confirmation, the status changes from Pending to Opted In.

Recap

In this exercise, you learned to implement and customize the Lite Registration flow.

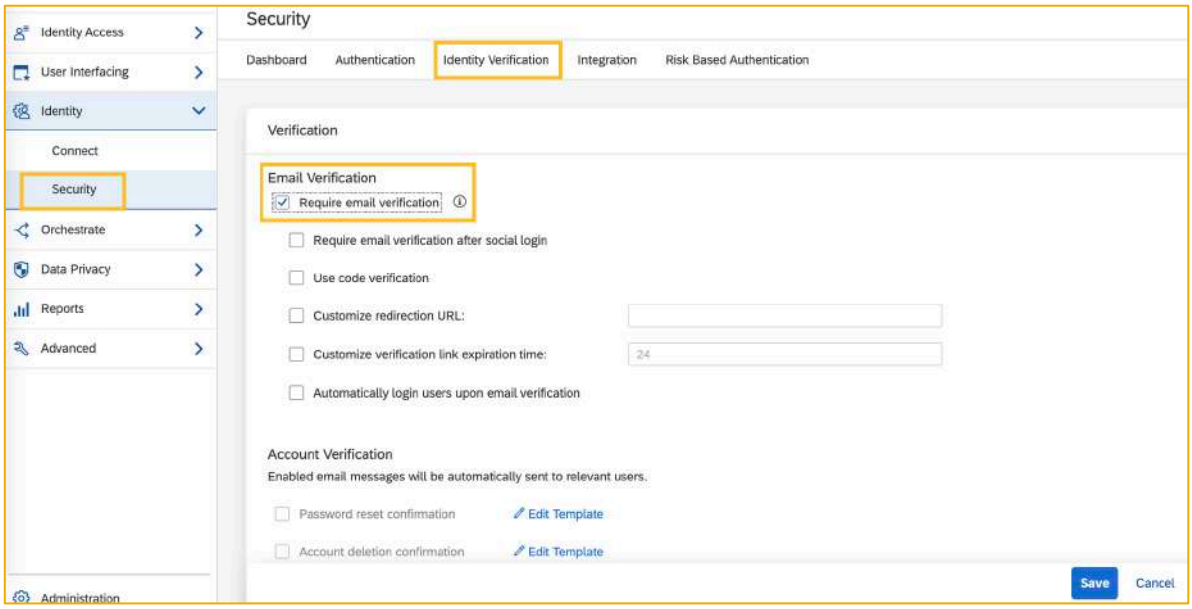
Task 3: Site Security Policies

Configure the site security policies to enable email verification for All Identities.

Solution

To configure the site security policies to enable email verification for All Identities, follow these steps:

1. On the left side navigation menu, select **Identity**, **Security** on the Security page.
2. Select the **Identity Verification** tab.



3. Select the **Require email verification** for all registrations.
4. Click on **Save** to save the policies configuration.
5. You can verify **Require email verification** settings once you register a new user with valid email ID. You will receive an email to verify.

Recap

In this exercise, you learned to use and customize the out-of-box Customer Identity screen-sets to perform different types of customer registrations and logins. You learned how to localize the screen-sets and change the existing user flows using conditions and the Web SDK.

Exercise 4: Full Registration

Full Customer Identity Implementation

Task 1: Full Customer Identity Implementation

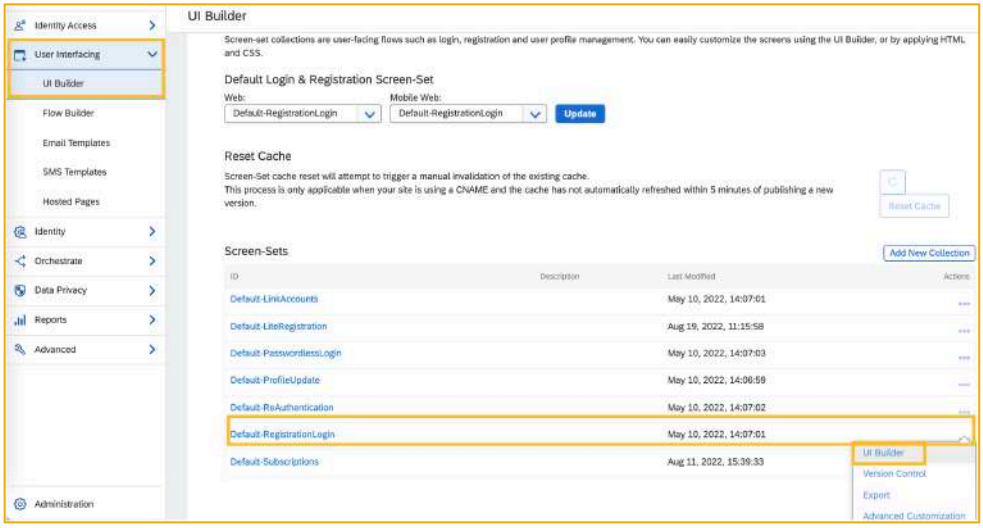
We will expand the previous exercise by adding full Screen-Sets-based registration and login support. Here's the list of requirements to implement full registration:

1. Screen-Set Default-RegistrationLogin, screens Registration and Login should allow only Facebook and Twitter as social providers. For now, just remove the other providers from those screens; you don't need to configure the actual social login functionality.
2. Insert a dropdown called Type of newsletter after the Subscribe to our newslette" checkbox. Map it to the new schema field data.typeOfNewsletter (you must first create the schema field). The possible choices for this field are *sports*, *politics,* or *music*. It should appear only when Subscribe to our newsletter is checked.
3. Open the index HTML file created in **Task 2** and add two links to it: Register and Login. When clicked, they should open the Default-RegistrationLogin Screen-Set's Registration and Login screens, respectively.
4. OPTIONAL: Create a Logout link that should call the logout Web SDK API endpoint. This link should be only visible if the user is currently logged in, so you must show after the page is loaded and there's a user currently logged in or after a successful login. You should hide the link after a logout.

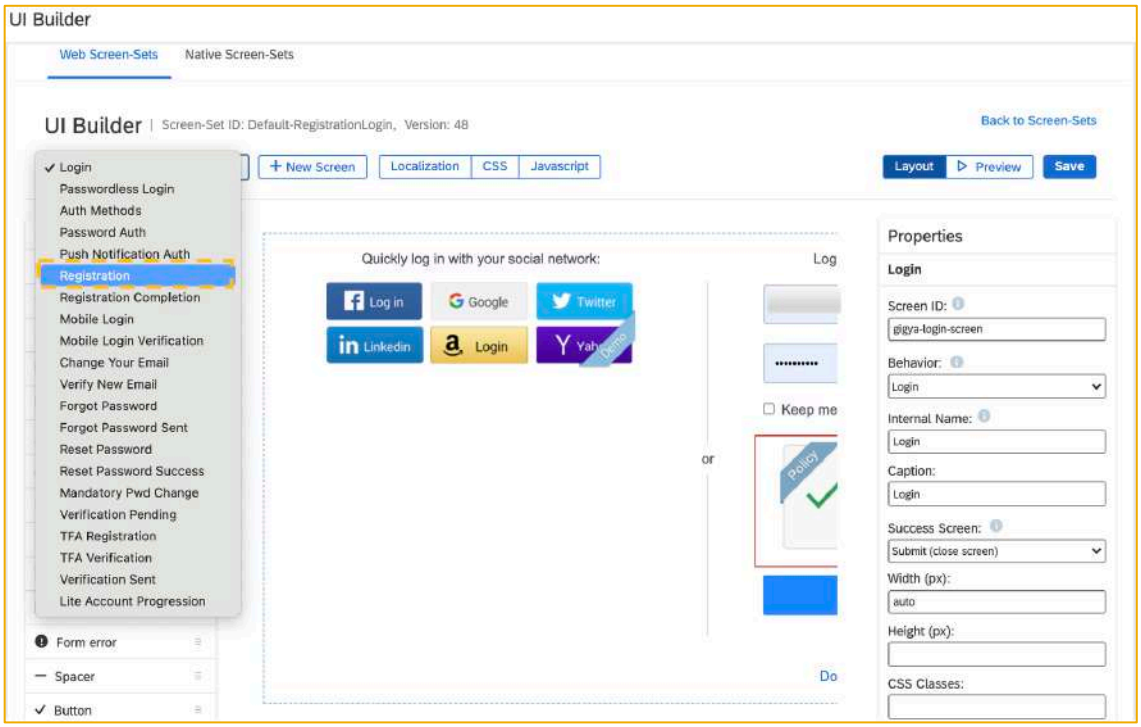
Solution

To complete a full Customer Identity implementation exercise, follow these steps:

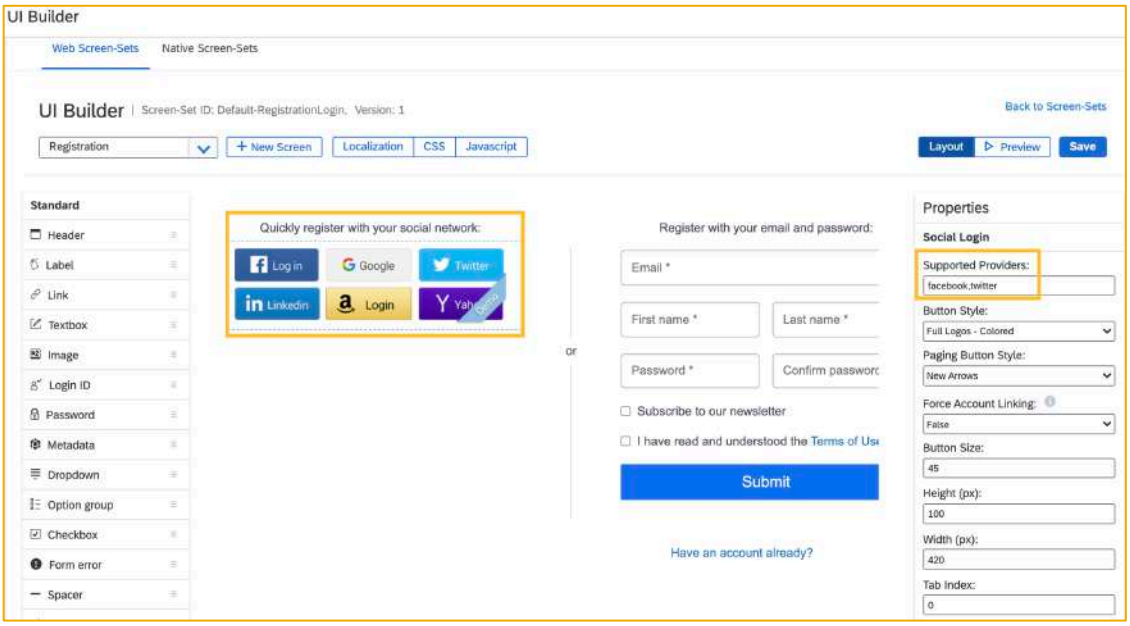
1. Access the **Customer Data Cloud Console** https://console.gigya.com
2. Select **one of the sites assigned to you**.
3. The Site Settings page opens up. On left side navigation menu select **User Interfacing** > **UI Builder**.
   - Under the Web Screen-Sets tab of the UI Builder page, hover over the **three-dot** actions menu of the Default-RegistrationLogin Screen-Set row.
   - Select **UI Builder**, or click on **Default-RegistrationLogin** to open the "Default-RegistrationLogin" screen-set.



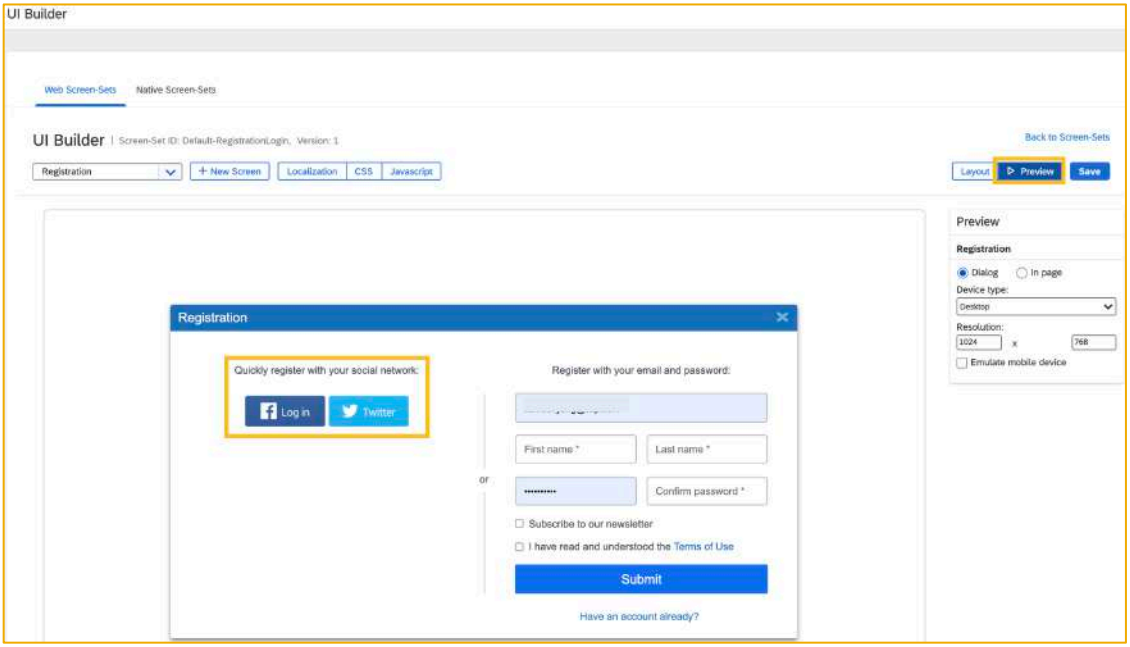
4. Select **Registration** screen from the screen dropdown.



5. Reduce the number of social networks available for social login to: **Facebook, Twitter**

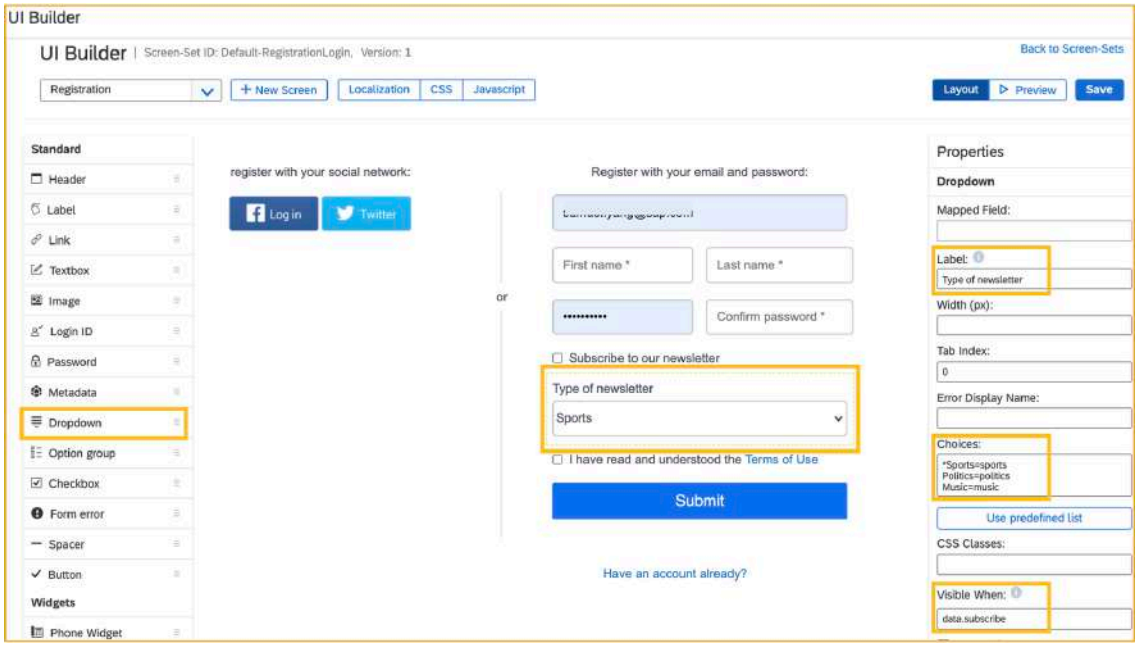


6. Click on **Preview** to see the changes.

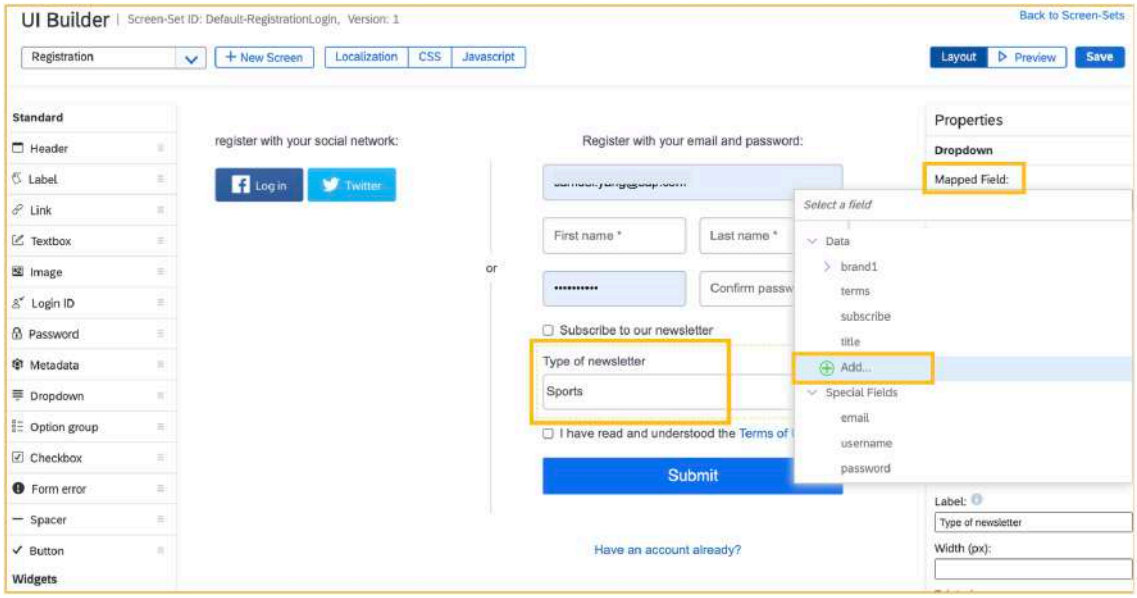


7. Go back to the **Layout** view.
   - Select **Login** screen from the screen dropdown on the left side of your screen
   - Repeat steps 5 and 7, to keep the Social Login experience between the Login and Registration screens consistent.

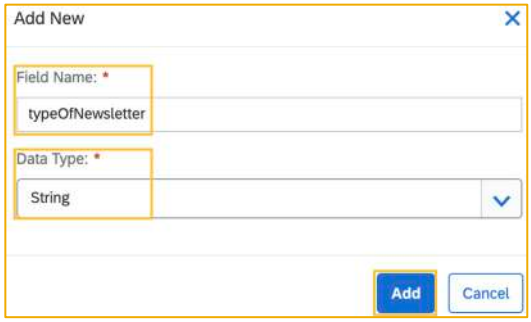
8. Click on **Dropdown** under Standard on the left panel.
   - Start a drag-and-drop operation into the area between the **Subscribe to our newsletter** and **I have read and understood the Terms of Use** checkboxes.



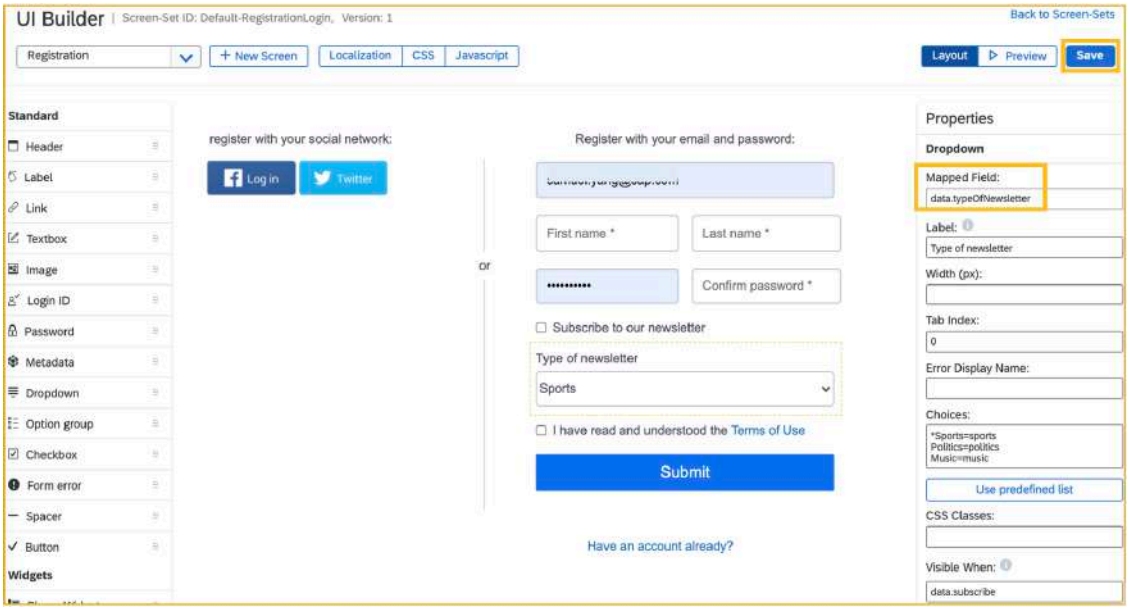
9. Map the new **Type of newsletter** field.
10. Click on **Mapped Field** value on the Properties section.
11. Click **Add** under the Data object.



12. When the Add New pop-up appears, enter **typeOfNewsletter** for Field Name.
13. Select the Data Type option as **String**.
14. Click **Add**.



15. Click **Save**.

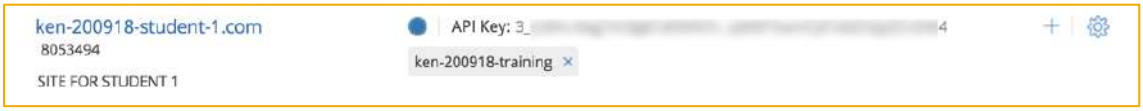


16. Modify the previously created file **index.html** in your GitHub repository and replace its contents by the HTML presented below.

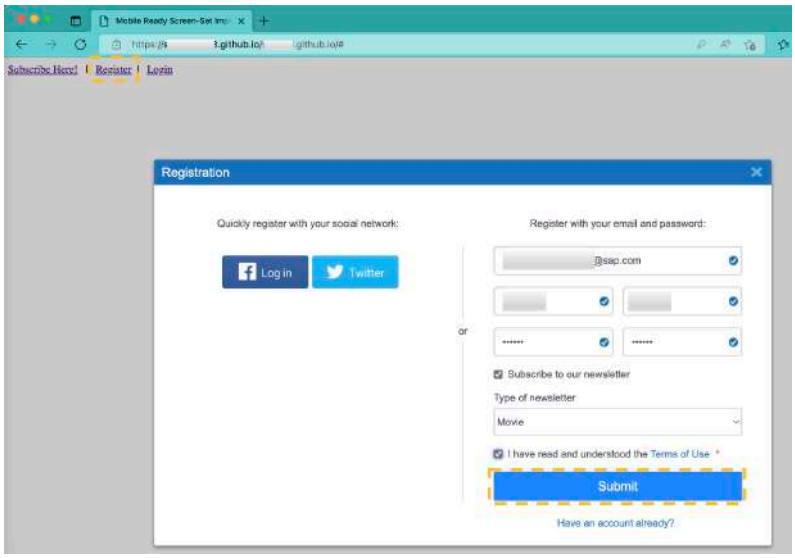
```
<html>

  <head>

    <title>Mobile Ready Screen-Set Implementation Example</title>

    <meta name="viewport" content="width=device-width">

    <SCRIPT type="text/javascript" lang="javascript"

      src="https://cdns.gigya.com/js/gigya.js?apikey=MY_API_KEY"></SCRIPT>

  </head>

  <body>

    <a href="#"
```

```
      onclick="gigya.accounts.showScreenSet({screenSet:'Default-
LiteRegistration'});">Subscribe Here!</a>   |  

    <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin', startScreen:'gigya-register-
screen'});">Register</a>  |  

    <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin'});">Login</a>

  </body>

</html>
```

17. Click **Save**.
18. Go to the **Customer Data Cloud Console** and copy the **API key of your site.**



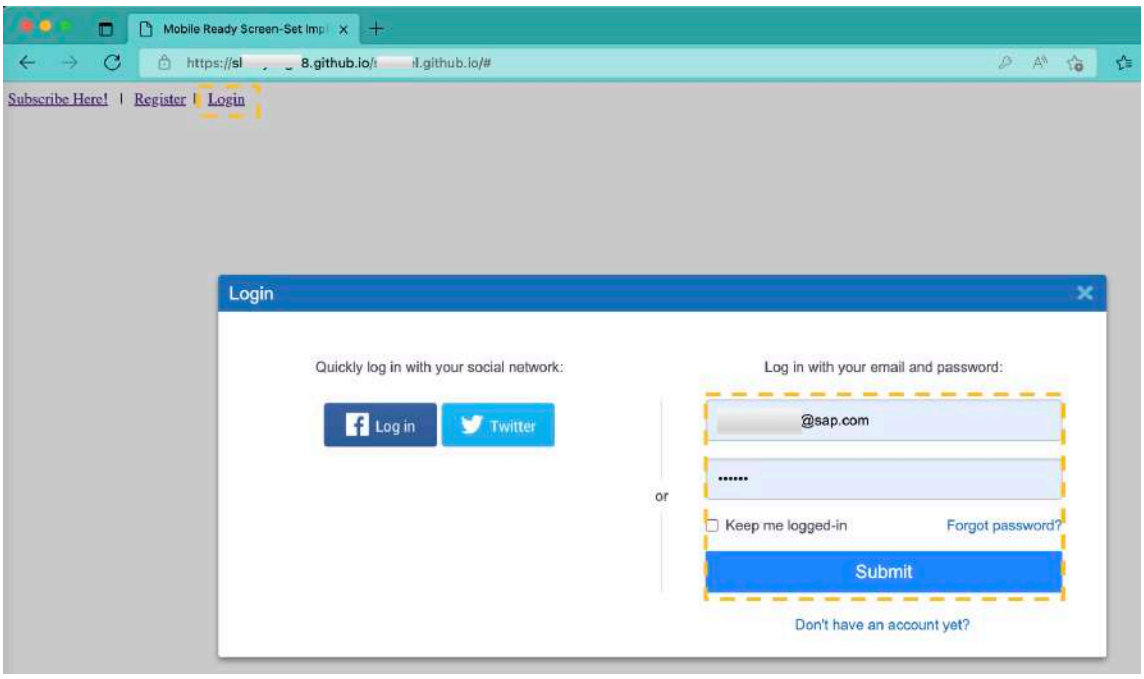
19. Go back to the **HTML file** and **paste the clipboard contents** to replace the MY_API_KEY placeholder at the end of line 5.
20. **Save** your file again.
21. Open the **html** file in the browser
    - Click on the **Register** link. The registration screen pops up, fill in fields:
    - **email** (use a valid email you have access to)
    - **first** and **last name.**
    - **password** and **password confirmation.**
    - Check the **Subscribe to our newsletter** box.
    - Select **one of the topics** on the Type of newsletter dropdown.
    - Select the **Terms of Use** box.
    - Click **Submit**.



22. Click **OK** to dismiss the Email Verification screen.

23. Check your email and activate the account.

24. Login to your newly registered account.



Recap

In this exercise, you learned how to edit a screen-set and remove/add a social network login option. You also learned how to add a Dropdown menu to the Login screen and how to map it to the Data field.

OPTIONAL STEPS

1. Let's now implement a **Logout** link.
   - Go to your **GitHub** repository,
   - Open **index.html**
   - Add **the following block of HTML** before the **</body>** tag

```
<span id="logout" hidden>

    |  <a href="#" onclick="logout();">Logout</a>

</span>

<script type="text/javascript" lang="javascript">

  // calls logout and then hides the logout link if the logout succeeds

  function logout() {

    gigya.accounts.logout({callback:function (response) {

      if (response.errorCode == 0) {

        document.getElementById("logout").hidden=true;

      }

      else {

        alert('Error:' + response.errorMessage);

      }

    }});

  }

</script>
```

2. When the page is loaded for the first time, it will show the **Logout** link if there's a user logged in. Paste the **following code** before the **</script>** tag

```
  // if there's a user logged in then shows the logout link

  gigya.accounts.getAccountInfo({callback:function (response) {

    if (response.errorCode == 0) {

      document.getElementById("logout").hidden=false;

    }

  }});
```

3. To show the **Logout** link after a login succeeds, paste **the following code** before the **</script>** tag.

```
    // traps the login event to show the logout link

      gigya.socialize.addEventHandlers({

        onLogin: function () {

          document.getElementById("logout").hidden=false;

        }

      });
```

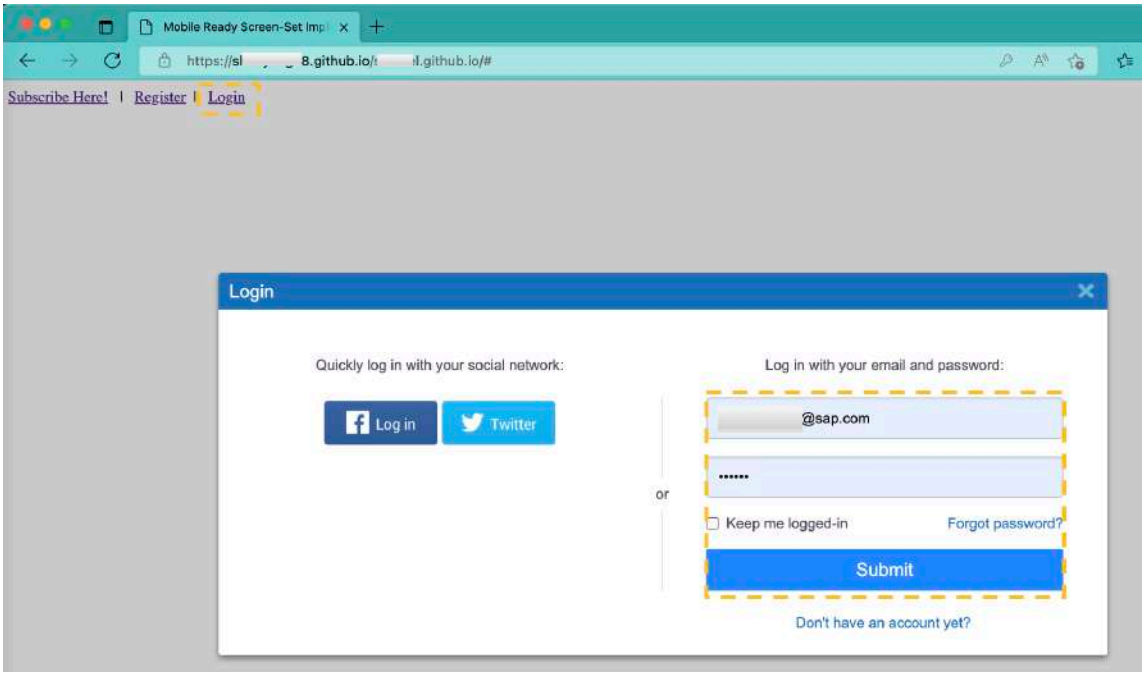
Your full **index.html** page source should look like this.

```
<html>

  <head>

    <title>Mobile Ready Screen-Set Implementation Example</title>

    <meta name="viewport" content="width=device-width">

    <SCRIPT type="text/javascript" lang="javascript"

      src="http://cdn.gigya.com/js/gigya.js?apikey=MY_API_KEY"></SCRIPT>

  </head>

  <body>

    <a href="#"

       onclick="gigya.accounts.showScreenSet({screenSet:'Default-
LiteRegistration'});">Subscribe Here!</a>  |  

    <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin', startScreen:'gigya-register-
screen'});">Register</a>  |  

    <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin'});">Login</a>

    <span id="logout" hidden>

        |  <a href="#" onclick="logout();">Logout</a>

    </span>

    <script type="text/javascript" lang="javascript">
```

```
    // calls logout then hides the logout link if the logout succeeds
    function logout() {
      gigya.accounts.logout({callback:function (response) {
        if (response.errorCode == 0) {
          document.getElementById("logout").hidden=true;
        }
        else {
          alert('Error:' + response.errorMessage);
        }
      }});
    }

    // if there's a user logged in then shows the logout link
    gigya.accounts.getAccountInfo({callback:function (response) {
      if (response.errorCode == 0) {
        document.getElementById("logout").hidden=false;
      }
    }});

    // traps the login event to show the logout link
    gigya.socialize.addEventHandlers({
      onLogin: function () {
        document.getElementById("logout").hidden=false;
      }
    });

  </script>
</body>
```

```
</html>
```

4. Now test the new logout functionality. Open the **html** page in the browser. Notice the **Logout** link is hidden.
5. Click on the **Login** link
6. Enter **credentials**.
7. Click **Submit**.



8. Notice that the **Logout** link appeared. Try refreshing the page once to check if it is still showing up.

9. Click on the **Logout** link, and check if the screen returns to the logged-out state.

Recap

In this exercise you learned how to implement a logout link to your site.

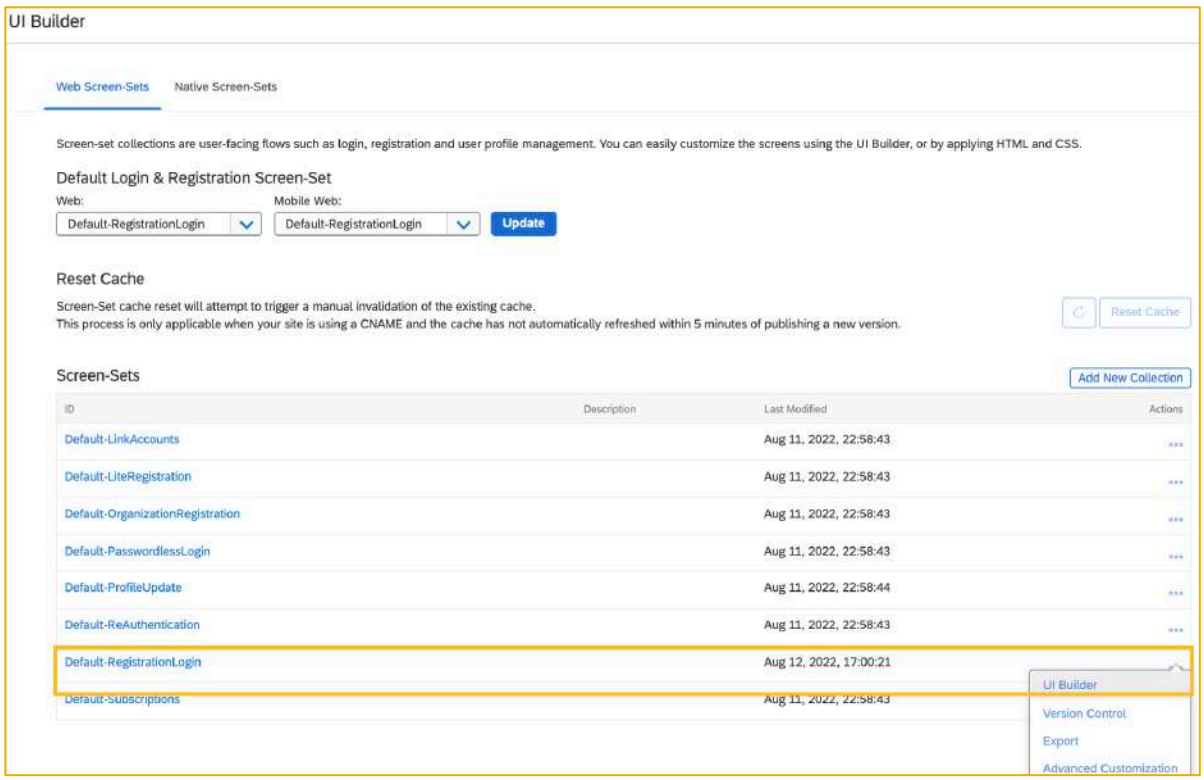
Task 2: Localization (OPTIONAL)

Translate your screen-sets for a different locale.

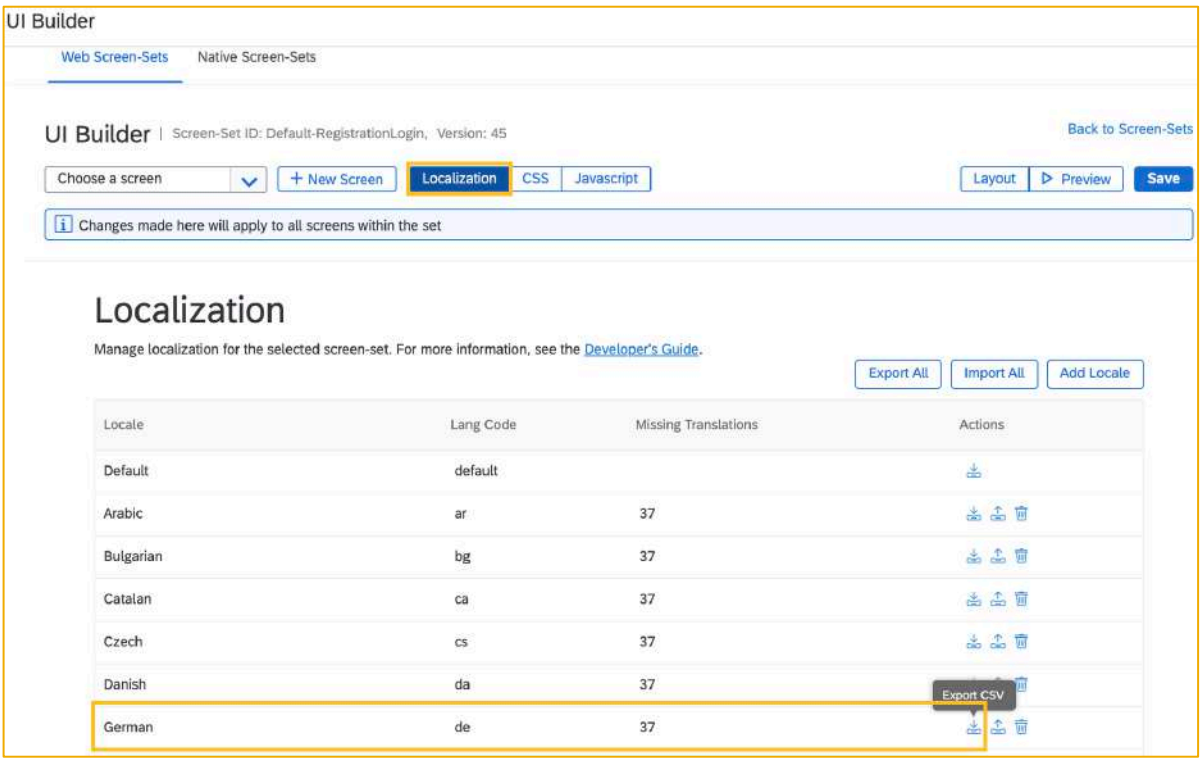
Solution

To update the screen-set labels for a different locale, follow these steps:

1. Access the **Customer Data Cloud Console** https://console.gigya.com
2. Select **one of the sites assigned to you**.
3. The Site Settings page opens up. On the left side navigation menu, select **User Interfacing > UI Builder** to open the UI Builder page.
4. Select the **Default-RegistrationLogin** under the Web Screen-Sets tab.



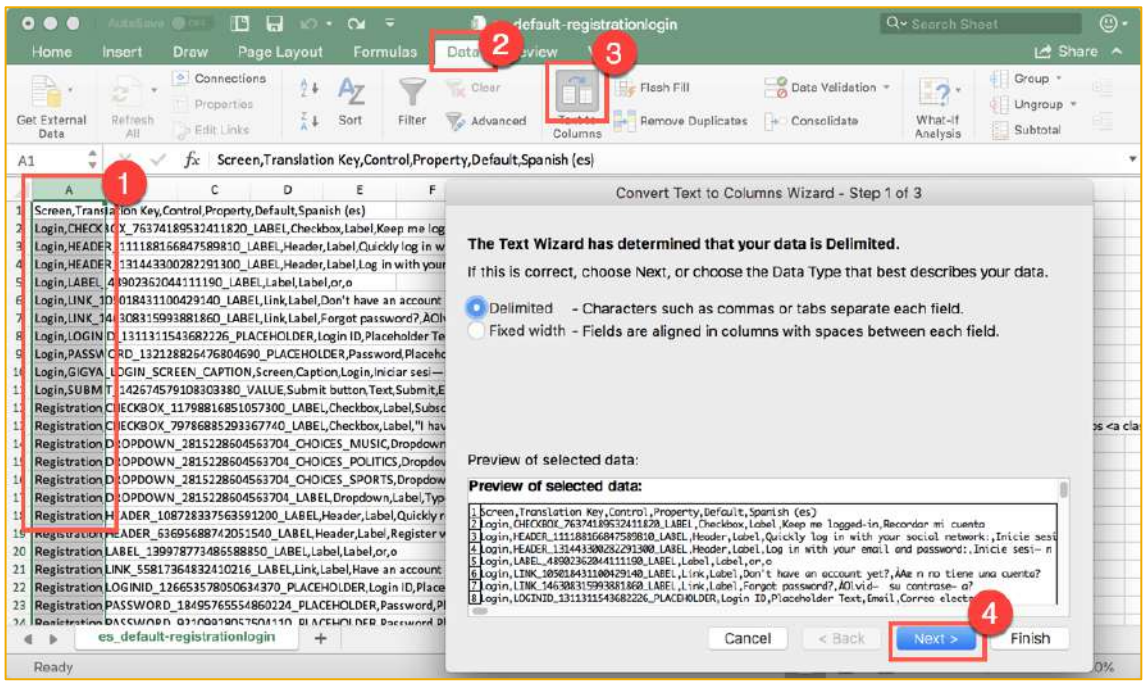
5. Click on the “Localization” button on the top of the screen.
6. When the “Localization” screen opens, you will notice some missing translations on the Missing Translations column. Let’s fix this.
   - Pick your language of choice.
   - Click on the **Export CSV** icon, located in the Actions column, to download its csv file.

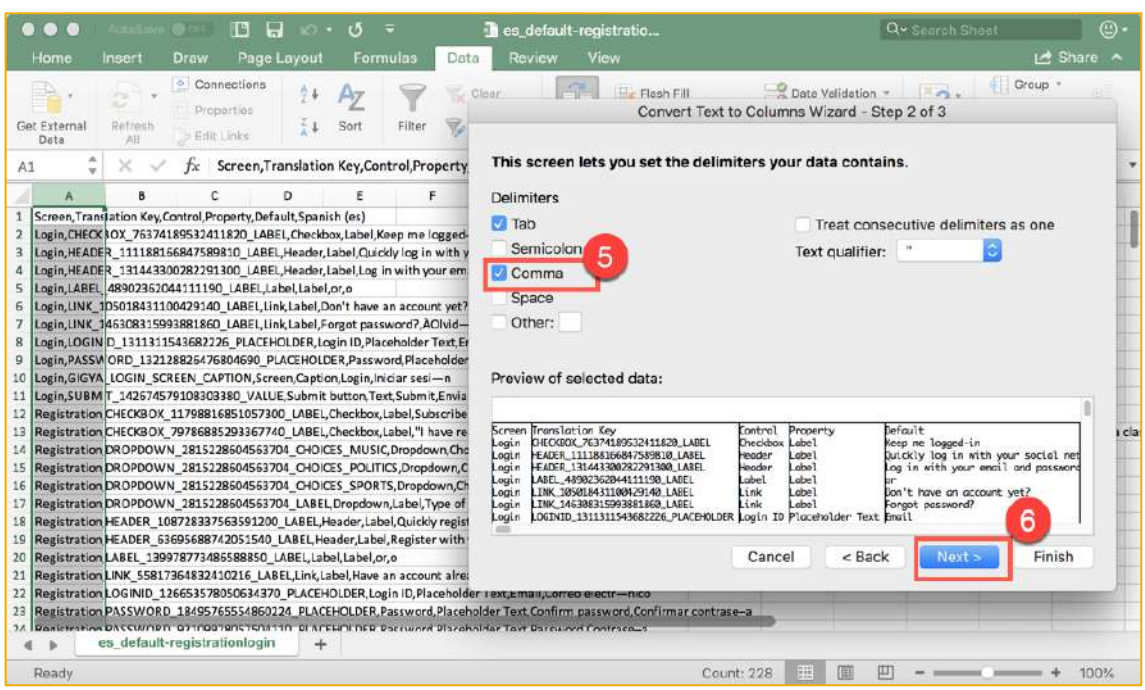


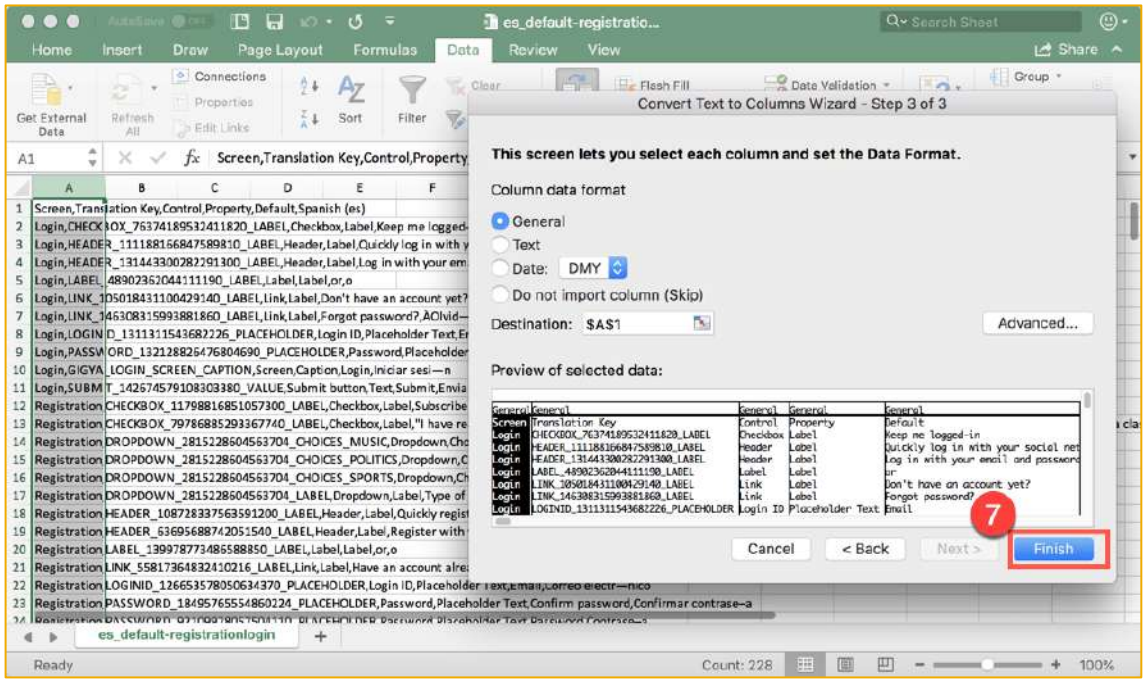
7. In the Full Registration Task, we added the Type of newsletter dropdown, and now we need to provide translation for the four missing labels: Type of newsletter, Music, Politics and Sports. Open the **downloaded CSV file**, locate the missing translations.

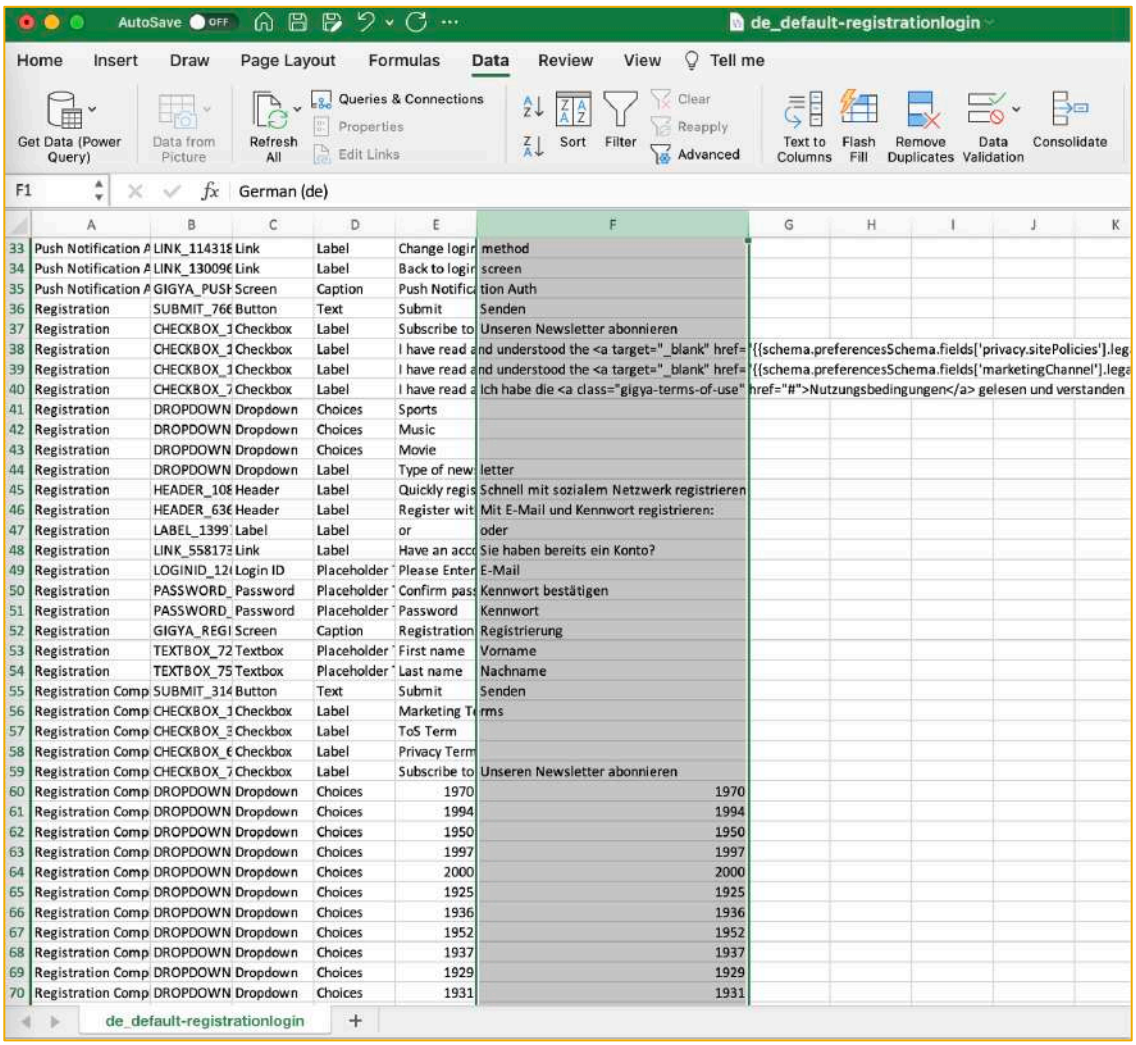
Note. Once you open the downloaded CSV file in Microsoft Excel, use the **Text to Columns** wizard with comma delimited to separate columns.

Follow the screenshots below.






es_default-registratio...
Convert Text to Columns Wizard - Step 3 of 3
This screen lets you select each column and set the Data Format.
Column data format
General
Text
Date: DMY
Do not import column (Skip)
Destination: A1
Advanced...
Preview of selected data:
Cancel
< Back
Next >
Finish
7
es_default-registrationlogin
Ready
Count: 228
100%

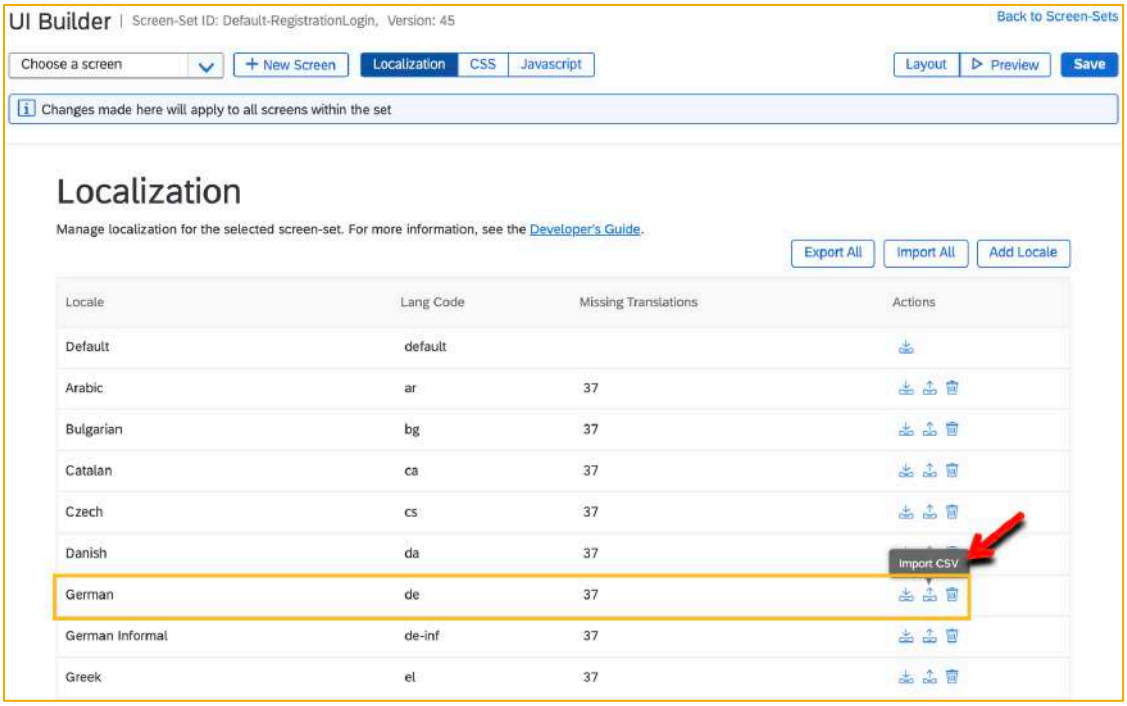

de_default-registrationlogin
Home
Insert
Draw
Page Layout
Formulas
Data
Review
View
Tell me
Get Data (Power Query)
Data from Picture
Refresh All
Queries & Connections
Properties
Edit Links
Sort
Filter
Clear
Reapply
Advanced
Text to Columns
Flash Fill
Remove Duplicates
Data Validation
Consolidate
F1
German (de)
de_default-registrationlogin

	A	B	C	D	E	F	G
1	Screen	Translation k	Control	Property	Default	German (de)	
2	Login	SUBMIT_142	Button	Text	Submit	Senden	
3	Login	CHECKBOX_7	Checkbox	Label	Keep me logged-in	Angemeldet bleiben	
4	Login	HEADER_111	Header	Label	Quickly log in with y	Schnell mit sozialem Netzwerk anmelden:	
5	Login	HEADER_131	Header	Label	Log in with your em	Mit E-Mail und Kennwort anmelden:	
6	Login	LABEL_4890	Label	Label	or	oder	
7	Login	LINK_105018	Link	Label	Don't have an accou	Haben Sie noch kein Konto?	
8	Login	LINK_146308	Link	Label	Forgot password?	Kennwort vergessen?	
9	Login	LOGINID_13	Login ID	Placeholder	Email	E-Mail	
10	Login	PASSWORD_	Password	Placeholder	Password	Kennwort	
11	Login	GIGYA_LOGI	Screen	Caption	Login	Anmelden	
24	Password Auth	SUBMIT_123	Button	Text	Submit		
25	Password Auth	HEADER_645	Header	Label	Please enter your password		
26	Password Auth	LINK_121888	Link	Label	Change login method		
27	Password Auth	LINK_685807	Link	Label	Back to login screen		
28	Password Auth	LINK_964979	Link	Label	Forgot password?		
29	Password Auth	PASSWORD_	Password	Placeholder	Password		
30	Password Auth	GIGYA_PASS	Screen	Caption	Password Auth		
31	Push Notification A	SUBMIT_521	Button	Text	Resend Push Notification		
32	Push Notification A	HEADER_442	Header	Label	Please approve notification in your mobile device		
33	Push Notification A	LINK_114318	Link	Label	Change login method		
34	Push Notification A	LINK_130096	Link	Label	Back to login screen		
35	Push Notification A	GIGYA_PUSH	Screen	Caption	Push Notification Auth		
36	Registration	SUBMIT_766	Button	Text	Submit	Senden	
37	Registration	CHECKBOX_1	Checkbox	Label	Subscribe to our nev	Unseren Newsletter abonnieren	
38	Registration	CHECKBOX_1	Checkbox	Label	I have read and understood the <a target="_blank" href="{{schema.preferenc		
39	Registration	CHECKBOX_1	Checkbox	Label	I have read and understood the <a target="_blank" href="{{schema.preferenc		
40	Registration	CHECKBOX_7	Checkbox	Label	I have read and und	Ich habe die <a class="gigya-terms-of-use" href="#">Nut	
41	Registration	DROPDOWN	Dropdown	Choices	Sports		
42	Registration	DROPDOWN	Dropdown	Choices	Music		
43	Registration	DROPDOWN	Dropdown	Choices	Movie		
44	Registration	DROPDOWN	Dropdown	Label	Type of newsletter		
45	Registration	HEADER_108	Header	Label	Quickly register wit	Schnell mit sozialem Netzwerk registrieren:	
46	Registration	HEADER_636	Header	Label	Register with your e	Mit E-Mail und Kennwort registrieren:	
47	Registration	LABEL_1399	Label	Label	or	oder	
48	Registration	LINK_558173	Link	Label	Have an account alr	Sie haben bereits ein Konto?	
49	Registration	LOGINID_12	Login ID	Placeholder	Please Enter Your E	E-Mail	

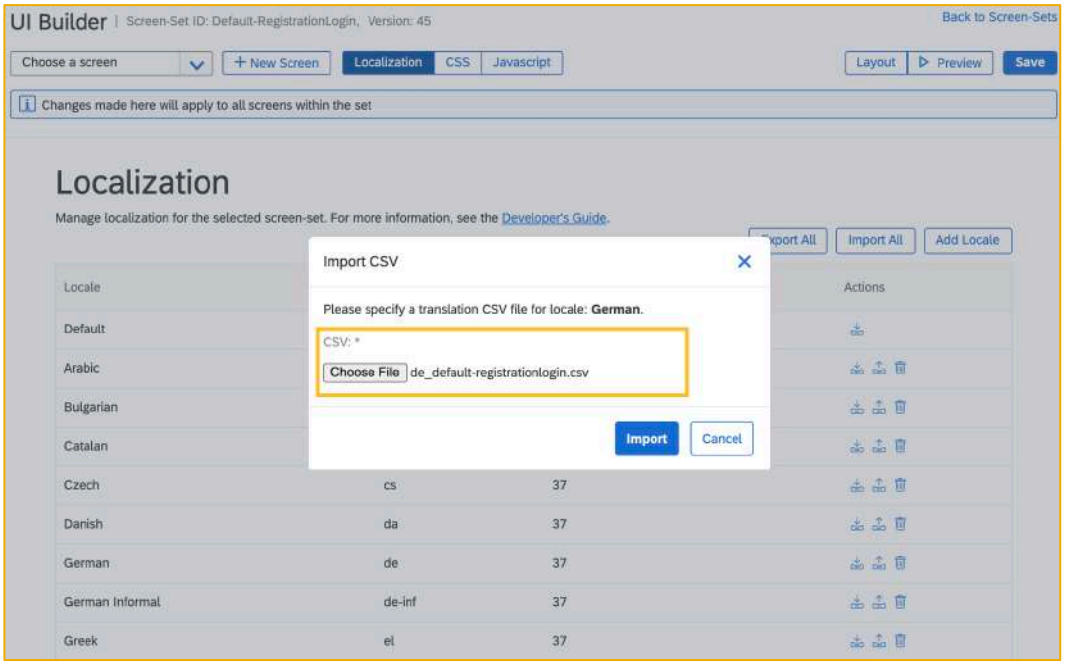
8. Provide the missing translations.
9. Save the file.

	A	B	C	D	E	F
1	Screen	Translation k	Control	Property	Default	German (de)
2	Login	SUBMIT_142	Button	Text	Submit	Senden
3	Login	CHECKBOX_7	Checkbox	Label	Keep me logged-in	Angemeldet bleiben
4	Login	HEADER_111	Header	Label	Quickly log in with y	Schnell mit sozialem Netzwerk anmelden:
5	Login	HEADER_131	Header	Label	Log in with your em	Mit E-Mail und Kennwort anmelden:
6	Login	LABEL_4890	Label	Label	or	oder
7	Login	LINK_105018	Link	Label	Don't have an accou	Haben Sie noch kein Konto?
8	Login	LINK_146308	Link	Label	Forgot password?	Kennwort vergessen?
9	Login	LOGINID_13	Login ID	Placeholder	Email	E-Mail
10	Login	PASSWORD_	Password	Placeholder	Password	Kennwort
11	Login	GIGYA_LOGI	Screen	Caption	Login	Anmelden
24	Password Auth	SUBMIT_123	Button	Text	Submit	
25	Password Auth	HEADER_645	Header	Label	Please enter your password	
26	Password Auth	LINK_121888	Link	Label	Change login method	
27	Password Auth	LINK_685807	Link	Label	Back to login screen	
28	Password Auth	LINK_964979	Link	Label	Forgot password?	
29	Password Auth	PASSWORD_	Password	Placeholder	Password	
30	Password Auth	GIGYA_PASS	Screen	Caption	Password Auth	
31	Push Notification A	SUBMIT_521	Button	Text	Resend Push Notification	
32	Push Notification A	HEADER_442	Header	Label	Please approve notification in your mobile device	
33	Push Notification A	LINK_114318	Link	Label	Change login method	
34	Push Notification A	LINK_130096	Link	Label	Back to login screen	
35	Push Notification A	GIGYA_PUSH	Screen	Caption	Push Notification Auth	
36	Registration	SUBMIT_766	Button	Text	Submit	Senden
37	Registration	CHECKBOX_1	Checkbox	Label	Subscribe to our nev	Unseren Newsletter abonnieren
38	Registration	CHECKBOX_1	Checkbox	Label	I have read and understood the <a target="_blank" href="{{scher	
39	Registration	CHECKBOX_1	Checkbox	Label	I have read and understood the <a target="_blank" href="{{scher	
40	Registration	CHECKBOX_7	Checkbox	Label	I have read and und	Ich habe die <a class="gigya-terms-of-use"
41	Registration	DROPDOWN	Dropdown	Choices	Sports	Sport
42	Registration	DROPDOWN	Dropdown	Choices	Music	Musik
43	Registration	DROPDOWN	Dropdown	Choices	Movie	Film
44	Registration	DROPDOWN	Dropdown	Label	Type of newsletter	Art des Newsletters
45	Registration	HEADER_108	Header	Label	Quickly register wit	Schnell mit sozialem Netzwerk registrieren:
46	Registration	HEADER_636	Header	Label	Register with your e	Mit E-Mail und Kennwort registrieren:

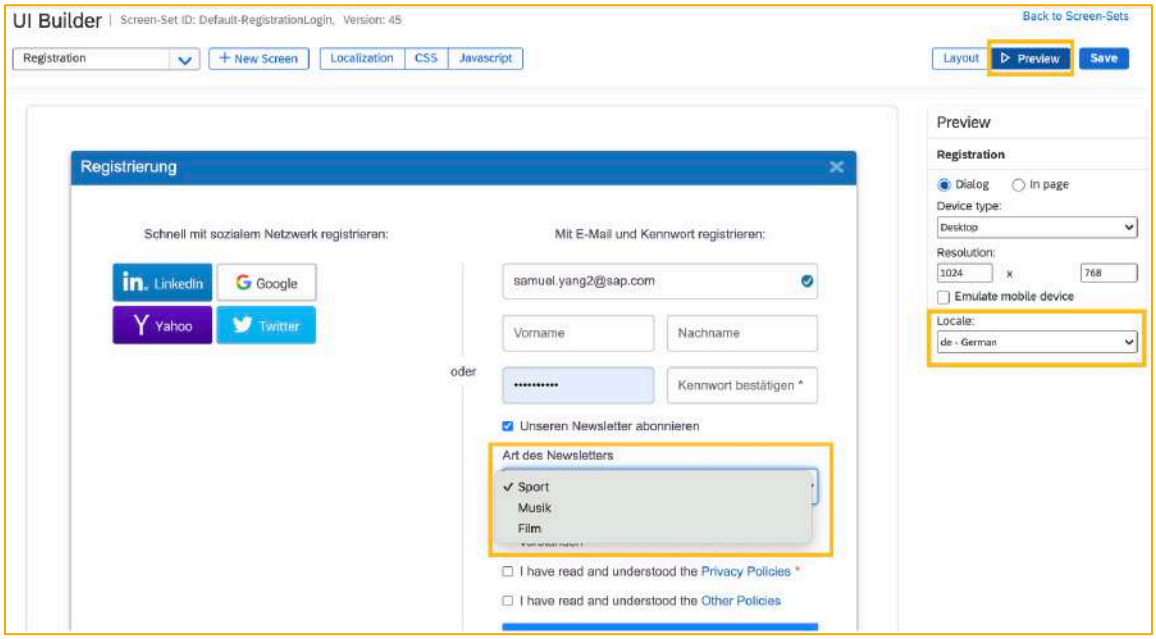
10. Go back to the console and click on the **Import CSV** icon, located in the same line as your target language, to upload your modified CSV file.



12. Click on **Choose File**.
13. Select your **modified CSV** file.
14. Click **Import**.

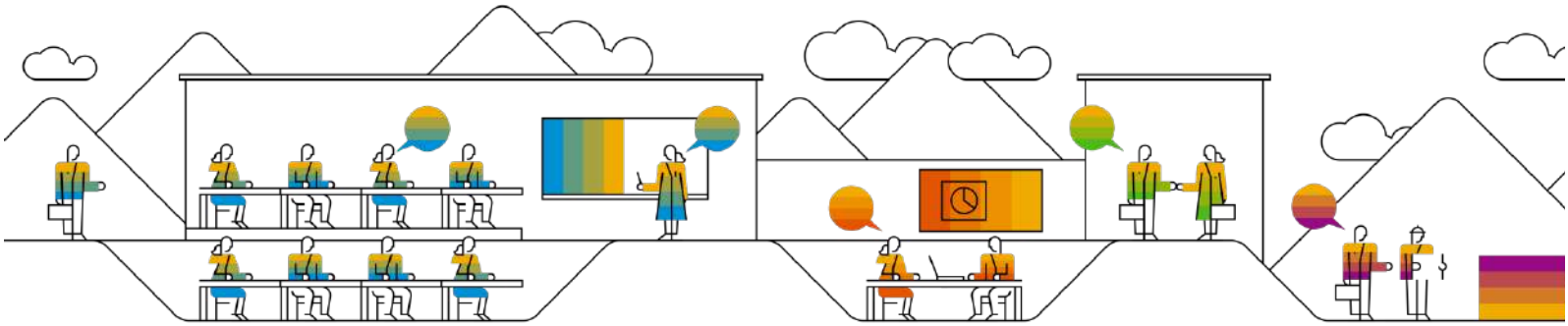


15. Let's now test our newly translated screen. We can test in the preview mode.
    - Change the locale to **German**
    - Verify that the language on the screen-set changes.



Recap

In this exercise you learned to add full Screen-Sets-based registration and login support.



Exercise 5: RESP API

API Requests

In this exercise you will access Customer Data Cloud REST API Tool to perform different API requests using your own API keys and secret keys. The purpose of the exercise is to learn how to access the Customer Data Cloud service through the REST interface. You will also perform some tasks on the Data Store.

Task 1: Perform Different API Requests

Add a new field in the schema for **data.CustomerType**.
Register 2 new users using Customer Data Cloud Rest API Console

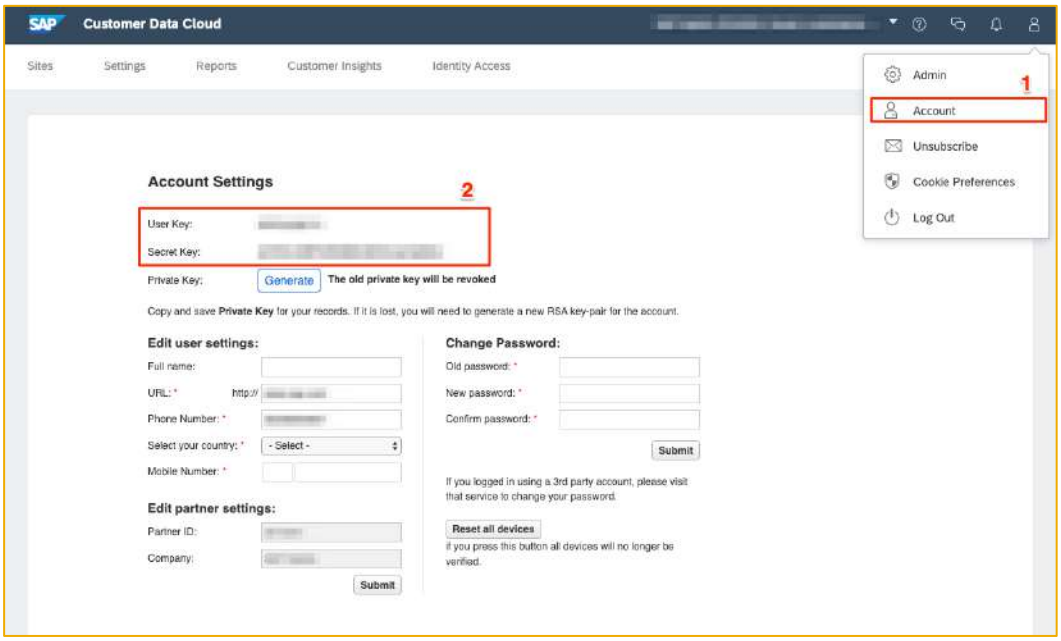
- User 1: Customer type Corporate" (string), 28 years old, Female, USA
- User 2: Customer Type Consumer" (string), 35 years old, Male, United Kingdom

Solution

To register a user using REST API, follow these steps:

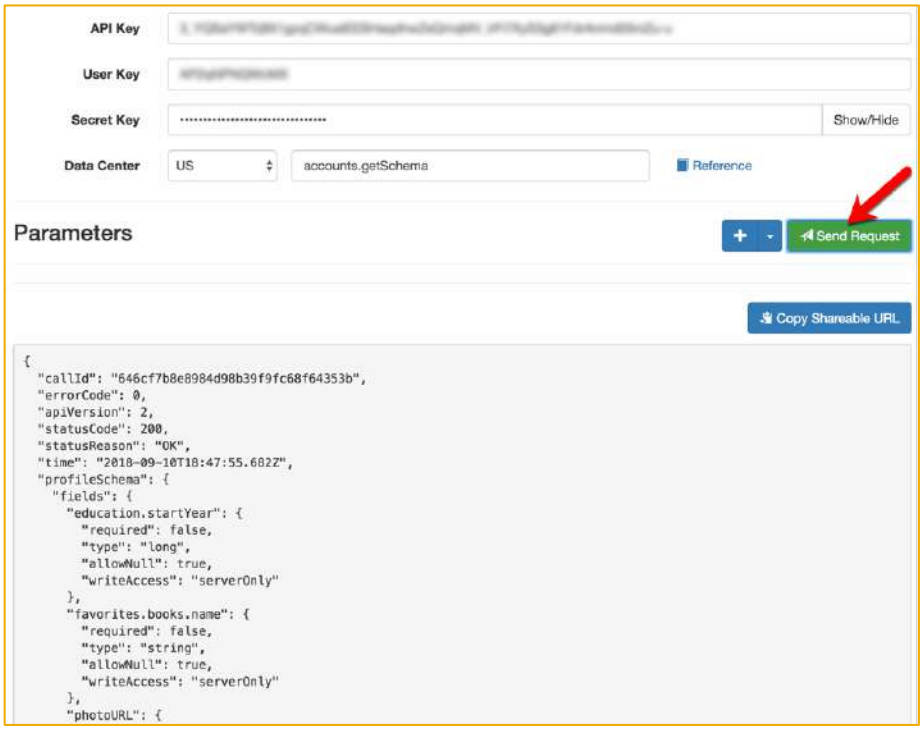
1. OPTIONAL: Let's take a look at the current schema. Access the API Tool (https://tools.gigya-cs.com/api/).
2. Call the **accounts.getSchema** Endpoint and fill the API Tool screen.
3. Click **Send Request**. Use **your site's API key** and **your user key** and **secret key**.

Note: For training purposes, use your personal user key and secret key.



Note: Remove the empty parameter fields. You don't need to send any parameters with this request.

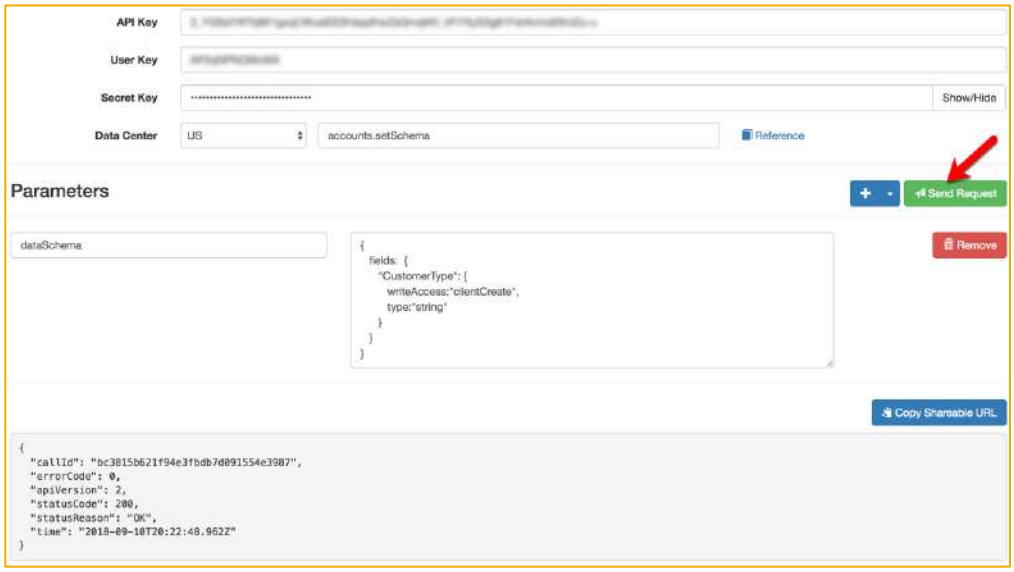
Make sure the payload returned from the call contains status Code 200.



The screenshot above is a partial picture of your current's site schema. ***Scroll down*** *to check all fields available on the schema.*

4. Perform a call to **accounts.setSchema,** passing the dataSchema parameter.
5. Fill the **API Key**, **User Key**, **Secret Key**, **Endpoint**, and **Parameters** as shown in the picture below.
6. Click the **Send Request** button.
7. Add a **Text-area input** for the dataSchema parameter which is more suitable for JSON values. Make sure the payload returned from the call contains the statusCode 200:

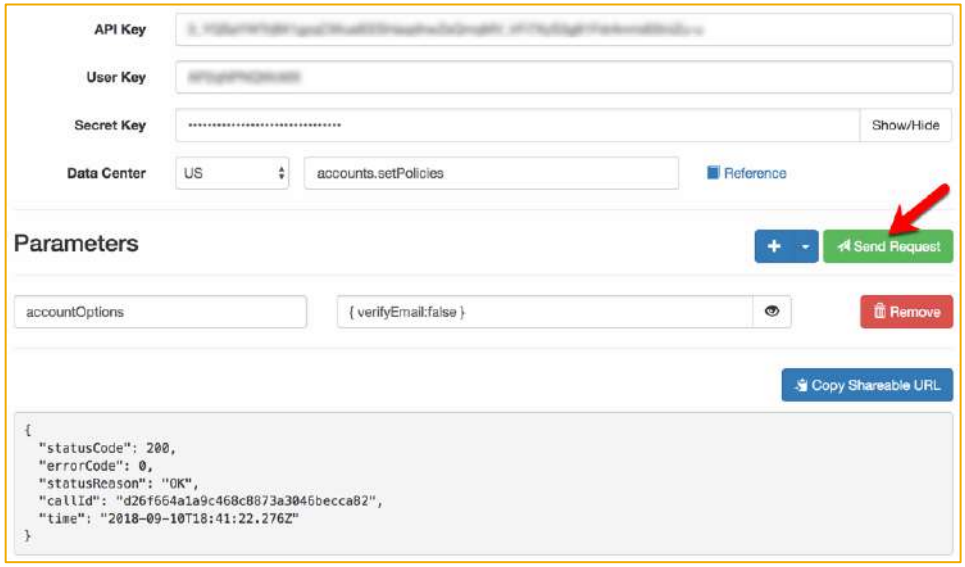
```
{
  fields:  {
     "CustomerType": {
        writeAccess:"clientCreate",
        type:"string"
      }
   }
}
```



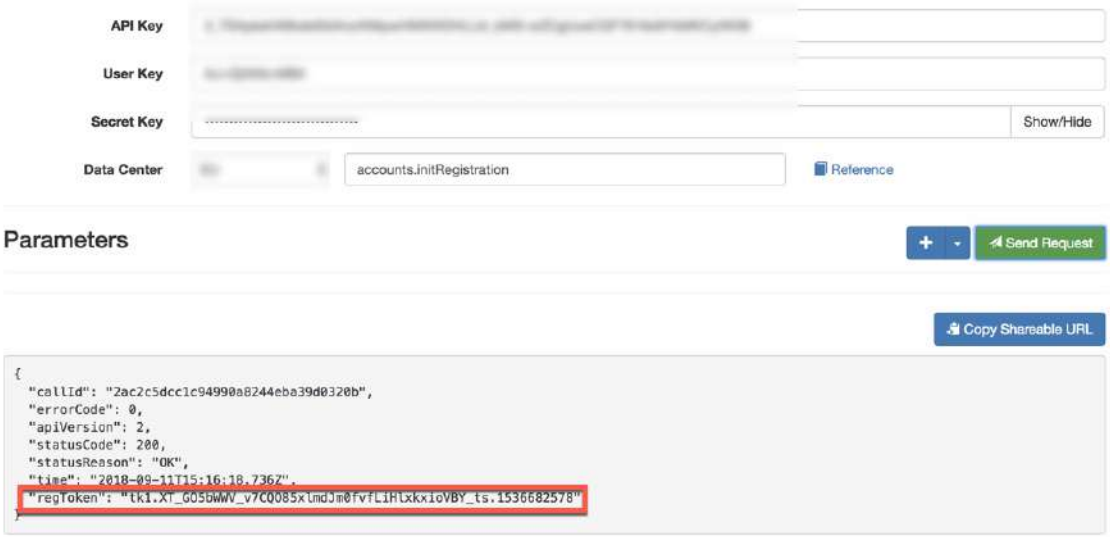
8. Create **two new users** using API.

   User 1: Customer Type **Corporate,** Age: **28**, Gender: **Female**, Country: **USA**
   User 2: Customer Type **Consumer,** Age: **35**, Gender: **Male**, Country: **United Kingdom**

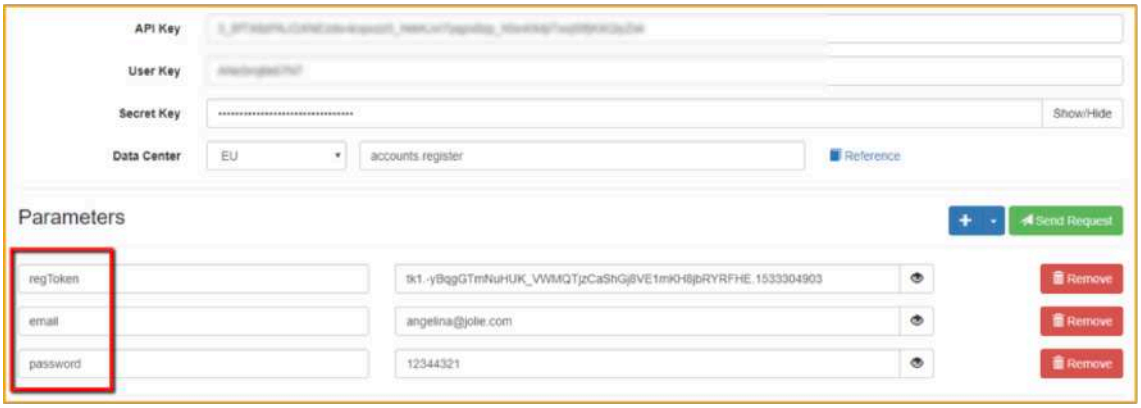
9. Continue to use the API tool to create the 2 new users shown above (https://tools.gigya-cs.com/api/).
10. Disable **email verification** by calling **accounts.setPolicies** and passing the **accountOptions** parameter with the value `{verifyEmail:false}`.
11. Click **Send Request** and look for a statusCode 200 inside the response payload.



12. Call the **accounts.initRegistration** endpoint to get the registration token.
13. Click **Send Request**. The response is a JSON code.



14. Make a note of the **regToken** value you received in the response to use in the next step.
15. Pass the value you just copied as the **regToken** parameter to the **accounts.register** API endpoint, along with the parameters **email** and **password**, set to whatever you want.
16. Click on the **plus** button below to add parameters to the request.



17. Click **Send Request**. Your response will contain an error message saying that *Registration was not finalized*. Don't worry about that, you will finalize registration in the next steps.
18. From the JSON code returned, collect the *new* **regToken**, which you'll need in the next step. Remember that the **accounts.register** endpoint returns a different **regToken** than did **accounts.initRegistration**.
19. Pass the new information to the ***setAccountnfo*** API endpoint to update the registered user with additional details.
20. Replace the value of **regToken** with the regToken value returned in the previous step.
21. Replace the **email** and **password** parameters with **profile** and **data**.

Use the following JSON code for the parameters.:

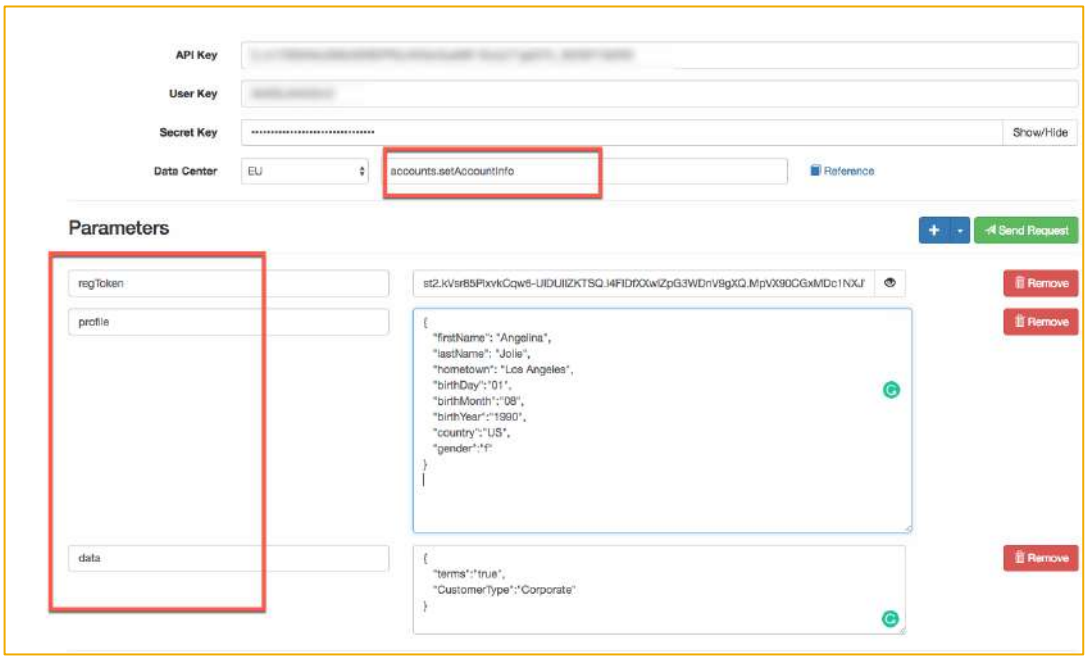
```
{
   "firstName": "Angelina",
   "lastName": "Jolie",
   "hometown": "Los Angeles",
   "birthDay":"01",
   "birthMonth":"08",
   "birthYear":"1990",
   "country":"US",
   "gender":"f"
}
```

For the **data** parameter:

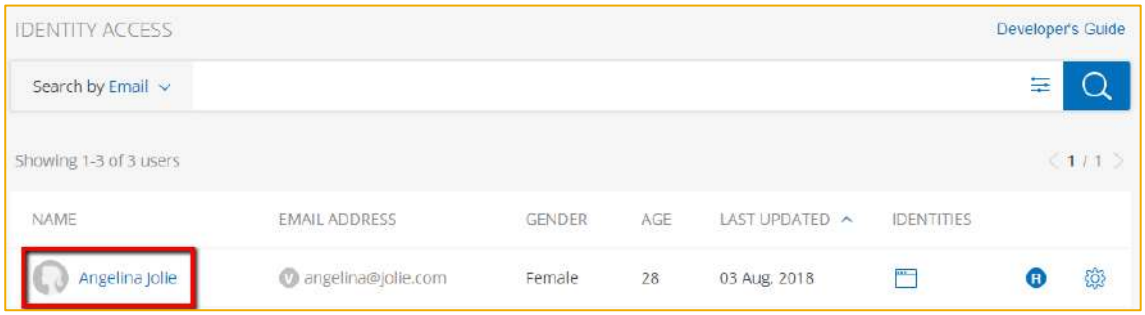
```
{
   "terms":"true",
   "CustomerType":"Corporate"
}
```

CustomerType *refers to the new field created in Step 1*.

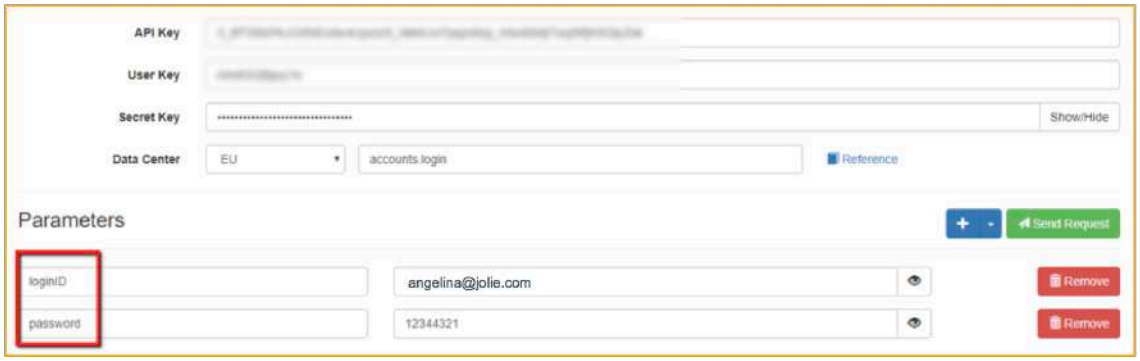
22. Click **Send the Request**.



23. Call the **finalizeRegistration** API endpoint to complete the user registration process. All we need to do is call the API with the right endpoint to confirm the account.
24. Remove all parameters except **regToken**. Use the **regToken** value received after calling the accounts.register endpoint.
25. Change the endpoint to **finalizeRegistration.**
26. Click **Send Request**. The account should now be created correctly. Verify in the console that the user exists:



27. Call the **login** API endpoint to login to the user account using the REST API and access the new user details.



28. Click **Send Request**.

The result should look like this:

```
{
  "callId": "6d0260fce6484ef2ac96c0746fcb14b0",
  "errorCode": 0,
  "apiVersion": 2,
  "statusCode": 200,
  "statusReason": "OK",
   ...
}
```

Repeat the steps above to create the second user.

- Customer Type **Consumer**
- Age: **35**
- Gender: **Male**
- Country: **United Kingdom**

Recap

In this exercise, you learned how to register a new user and update an existing customer profile using Customer Data Cloud REST API endpoints.

Task 2: Data Store

Create an array of family members using the REST API tool. Each member should have a **name**, **relationship**, and **email**. Parameters must belong to the DS and are created using **ds.setSchema**. The following JSON code can be used as mock-up to help on this step:

```
{
   “fields”: {
      “make”: {
         “writeAccess”: “clientModify”,
         “type”: “string”
      },
      “model”: {
         “writeAccess”: “clientModify”,
         “type”: “string”
      },
      “color”: {
         “writeAccess”: “clientModify”,
         “type”: “string”
      }
   }
}
```

Store information for two family members in this schema using **ds.store**:

- Betty Mother, betty@gmail.com
- Frank Father, frank@gmail.com

Parameters required:

- **Data**
- **Type** (whatever you set it as)
- **OID** (random number you come up with)

1. Search the users and their details using the API.

Solution

To perform actions on the Data Store using the REST API, follow these steps:

1. Create the parameters in DS using **ds.setSchema**
2. Access the **REST API** tool and enter the **authentication attributes** (API key, User key and Secret).
3. Create two new parameters:
   - One must be named **type** and contain the name of the schema, in this case: **family**.
   - The second parameter must be named **dataSchema** and contain the following JSON code:

```
{
   "fields": {
      "name": {
         "writeAccess": "clientModify",
         "type": "string"
       },
      "relationship": {
         "writeAccess": "clientModify",
         "type": "string"
      },
      "email": {
         "writeAccess": "clientModify",
         "type": "string"
      }
   }
}
```

OPTIONAL: If you want to add a regular expression to validate the mail as a real mail format, add the following string to the **email** field (anywhere is fine but must be inside the **{ }** of the attribute, for example between the attributes *writeAccess* and *type*):

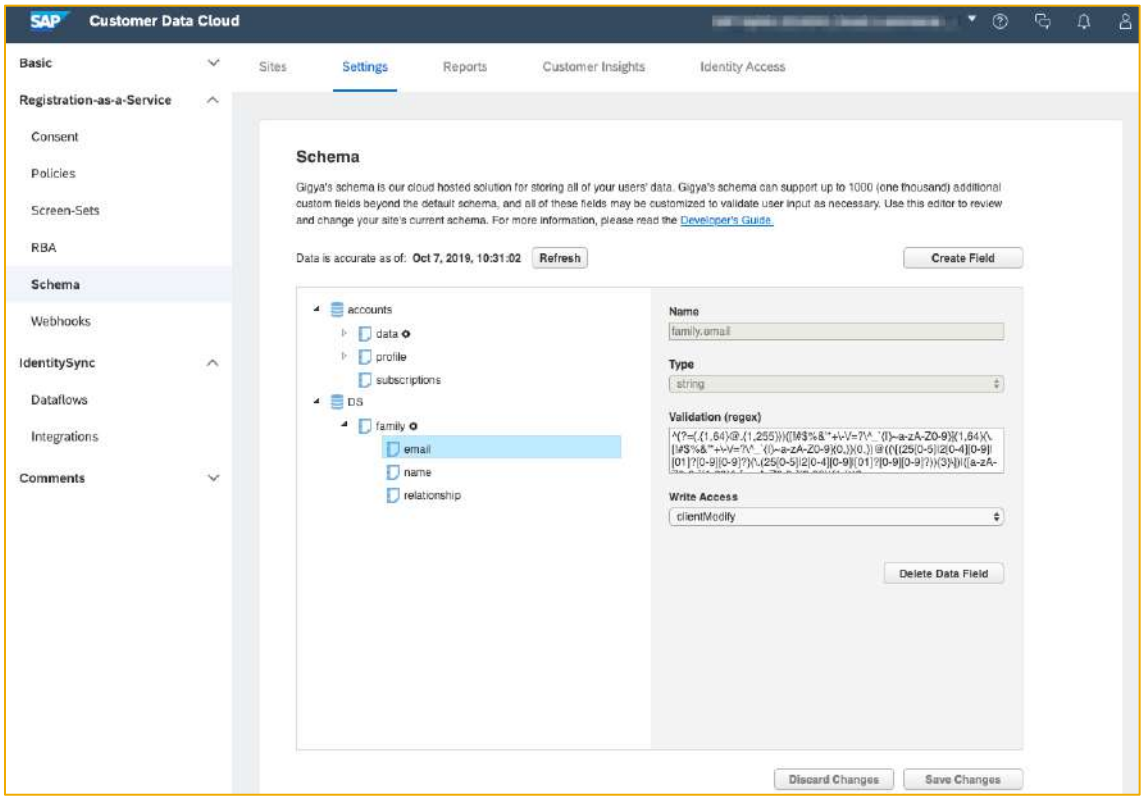
```
„format“ : „regex(,^(?=(.{1,64}@.{1,255}))([!#$%&‘*+\-\/=?\^_`{|}~a-zA-Z0-9}]{1,64}(\.[!#$%&‘*+\-\/=?\^_`{|}~a-zA-Z0-9]{0,}){0,})@((\[(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}\])|([a-zA-Z0-9-]{1,63}(\.[a-zA-Z0-9-]{2,63}){1,}))$‘)“,
```

Regular expression should validate all type of mails.

Note: Best practice is to validate any field that requires validation on the client side. This can be used to enforce the validation.

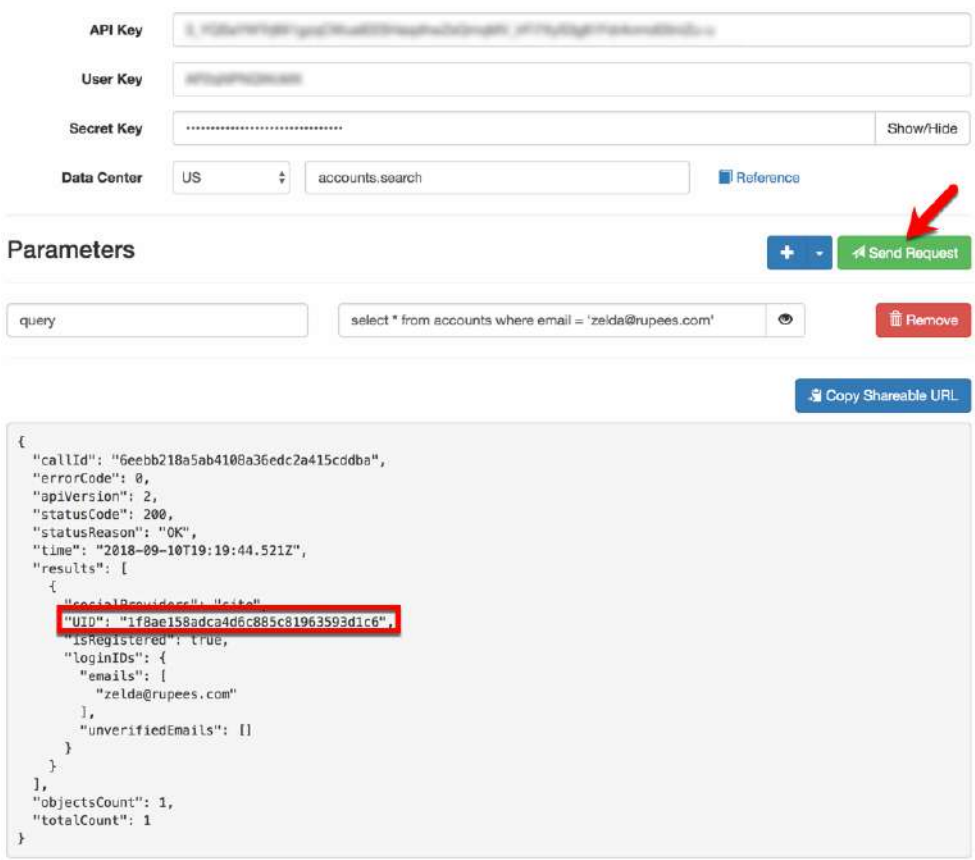
4. Click on the **Send Request** button on the API tool.
5. Access the console and check the schema modifications.

   Something like the following should be shown:



6. Add **two new family members** and test if everything works correctly.
7. Access the **REST API** tool and provide the **authentication attributes** needed for the REST call.
8. We need the UID of a user in order to associate them with the information about their Mother and Father. Search for an **existing user by their email** using the Endpoint **accounts.search.**
9. Set the query parameter **to** `select * from accounts where email = 'zelda@rupees.com'` (replace zelda@rupees.com with an existing user email in your site's accounts database.)
10. Click **Send Request**. Make sure that the statusCode returned is 200.
11. Then, select and copy the **UID** value.

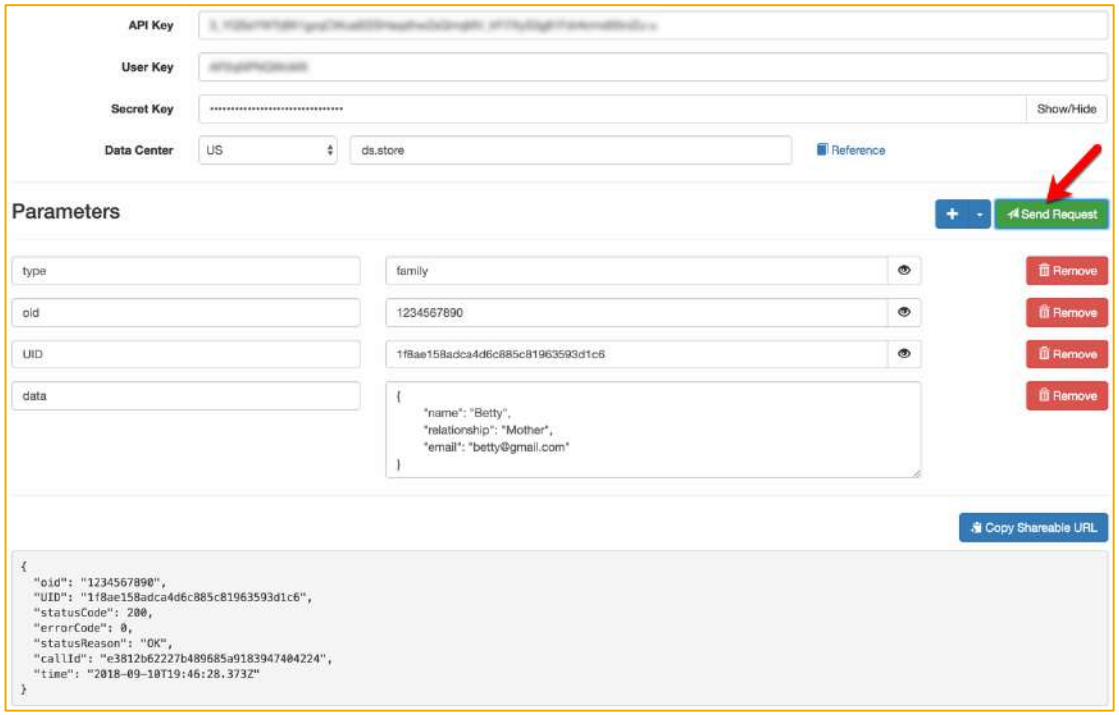
    The image below was cropped, you must have to scroll down until you find the UID value



12. Change the endpoint to **ds.store**,
13. Change the previous parameters
    - type set to the name of the schema, in this case ***family***
    - OID set to a **unique random number**, for example 1234567890
    - UID set to the **UID value you copied** in the previous step
    - data set to the **following JSON**:

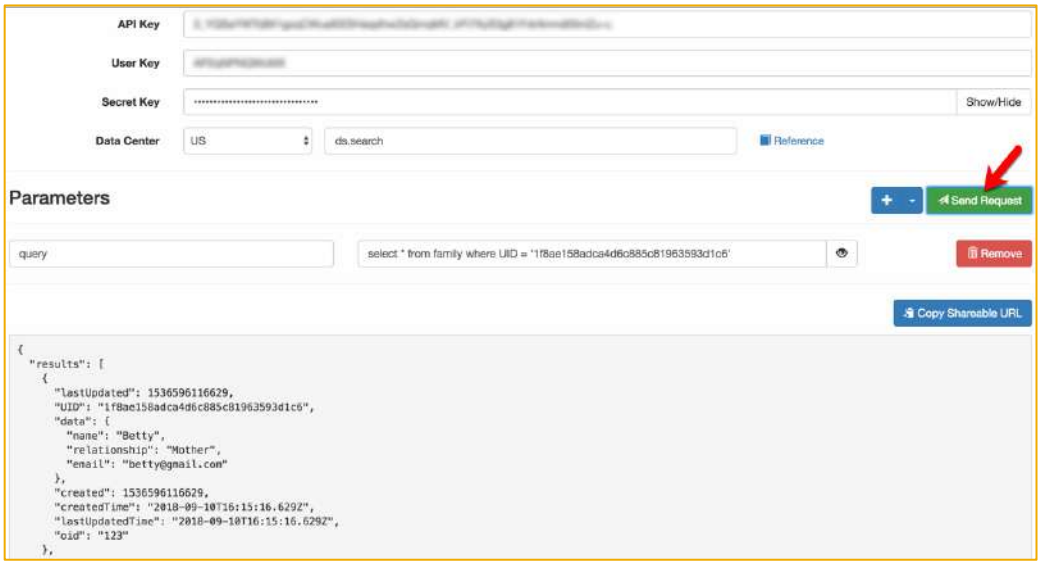
```
{
    "name": "Betty",
    "relationship": "Mother",
    "email": "betty@gmail.com"
}
```

14. Click **Send Request**. Make sure the resulting status code is 200.



.

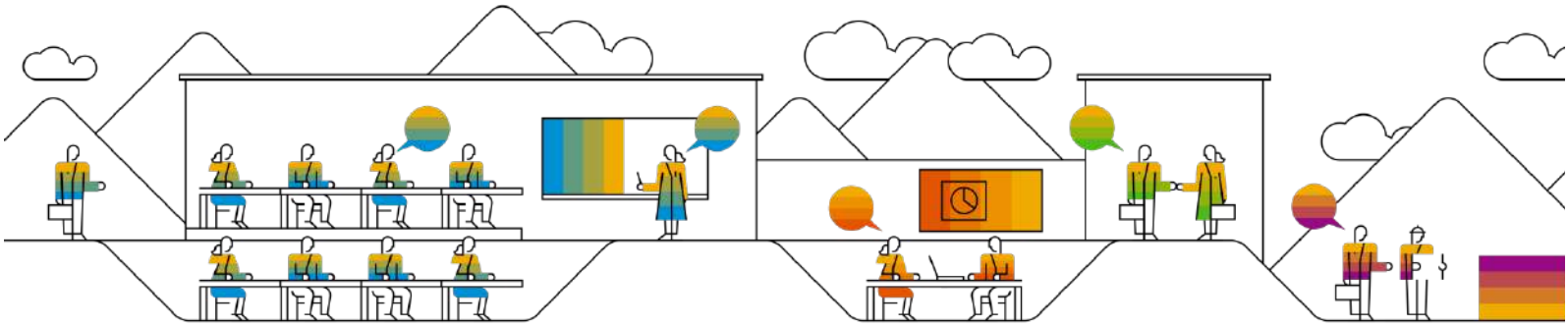
1. Repeat this step for the other family member (Frank, the father).
2. Perform a query Search to search for the user and their details.
   - The endpoint for this operation is **ds.search**.
   - The query parameter should be " `select * from family where UID = '<UID value from previous step>'`.
3. Click **Send Request**. Make sure the result payload has the family members added previously.



Recap

In this exercise you learned how to register a new user using Customer Data Cloud REST API endpoints and update some details.

In exercise 2 you performed actions on the Data Store using the REST API.



Exercise 6: Enterprise Consent and Preference Management (ECPM)

Implement Customer Consent

In this exercise, you will implement SAP Customer Consent. This exercise covers how to create and configure a new consent statement, manage versioning, and add it to a screen-set using the UI Builder. And finally, you will check the consent records in the consent vault.

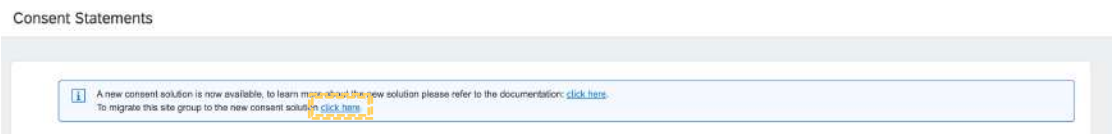
Task 1: Create Consent statements.

Create a Consent statement which should have Privacy Policy and Terms of Service and optionally one Other Consent Statement in SAP Customer Cloud Console.

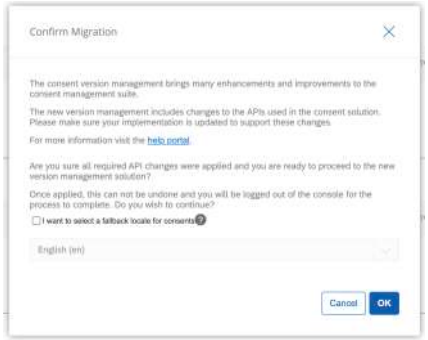
Solution

To implement SAP Customer Consent, follow these steps:

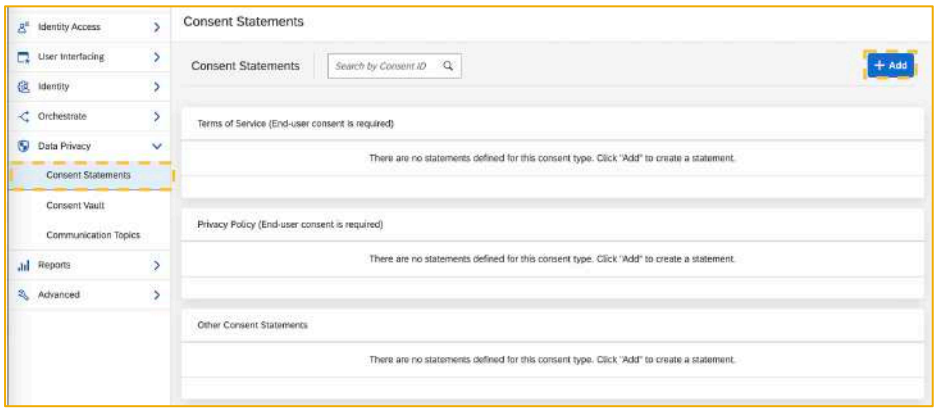
1. Open the **CDC site assigned to you**, and navigate to **Data Privacy -> Consent Statements**
2. The system asks if you want to migrate to the new consent solution (Consent v2). Click on the link to migrate to Consent v2.



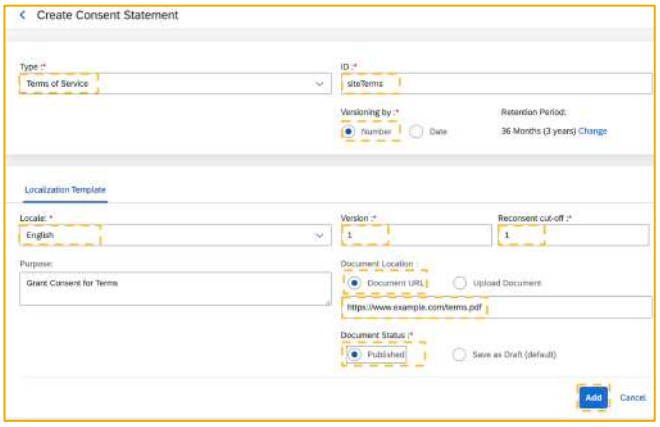
3. Click OK to confirm the migration.



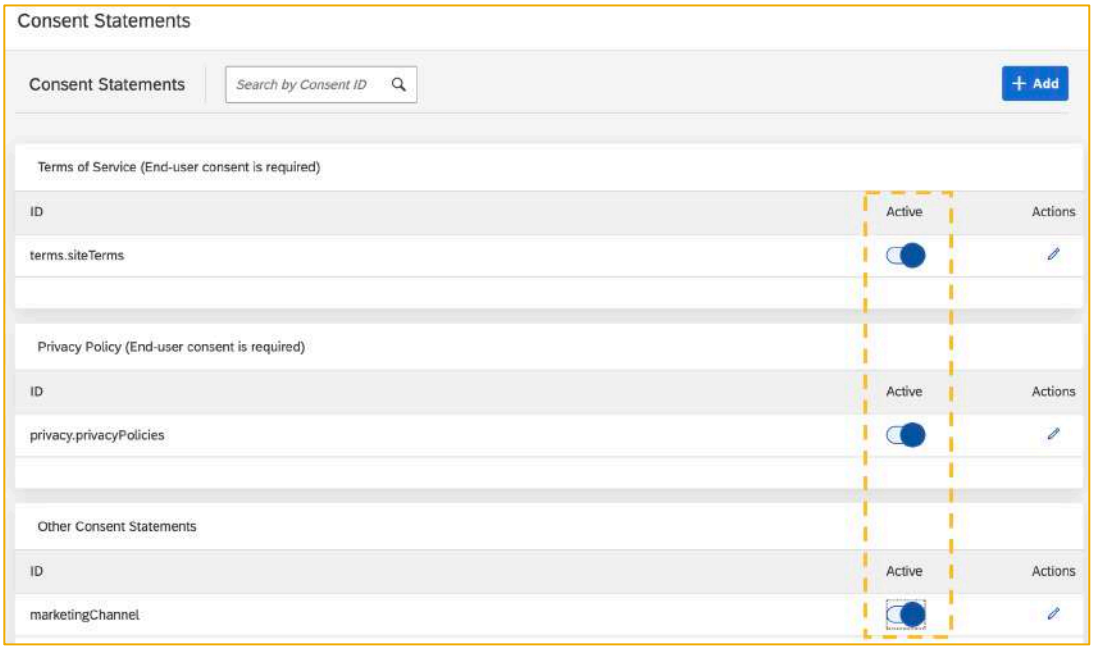
4. Click on **+ ADD**.



5. In Create Consent Statement, enter the following:
   - Type field: select **Terms of Service**.
   - ID field: enter a unique name for your *Terms of Service* consent statement. In this exercise, we will use **siteTerms**.
   - Versioning by: change to Number, which is the default versioning scheme.
   - Version: enter the number "**1**", indicating that this is the first version of the consent document.
   - Re-consent cut-off: enter the number "**1**",
   - Document Location: select the **Document URL** to link the terms document to the consent.
   - URL textbox: enter the **https link to the terms document**, e.g., https://www.example.com/terms.pdf.
   - Document Status: check the **Published** option to finalize the consent statement creation or editing. Check the **Save as Draft (default)** option to put it in draft status and later on you can further edit it.



6. Repeat the above steps to create **Privacy Policy** and **Other Consent** Statements.
   - Use **privacyPolicies** and **marketingChannel** as ID values.
   - **Activate** all the three consents.



Task 2: Add Consent Statements to Screen-set.

Take the Consent statements you created in Task 1 and add them to a Screen-set.

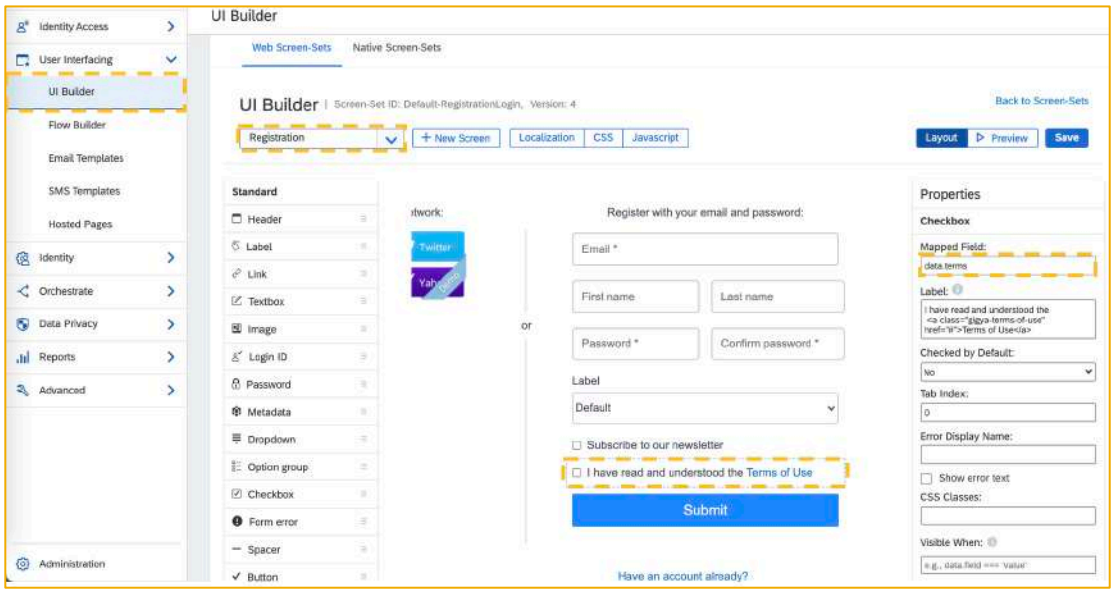
HINT: Add a checkbox to the registration screen-set and map the checkbox to the isConsentGranted property of the consent schema field.

Note: Any consent statement that is set to *active* for your site must be included in your registration and registration completion screens, otherwise users will not be able to complete their registration or provide re-consent.

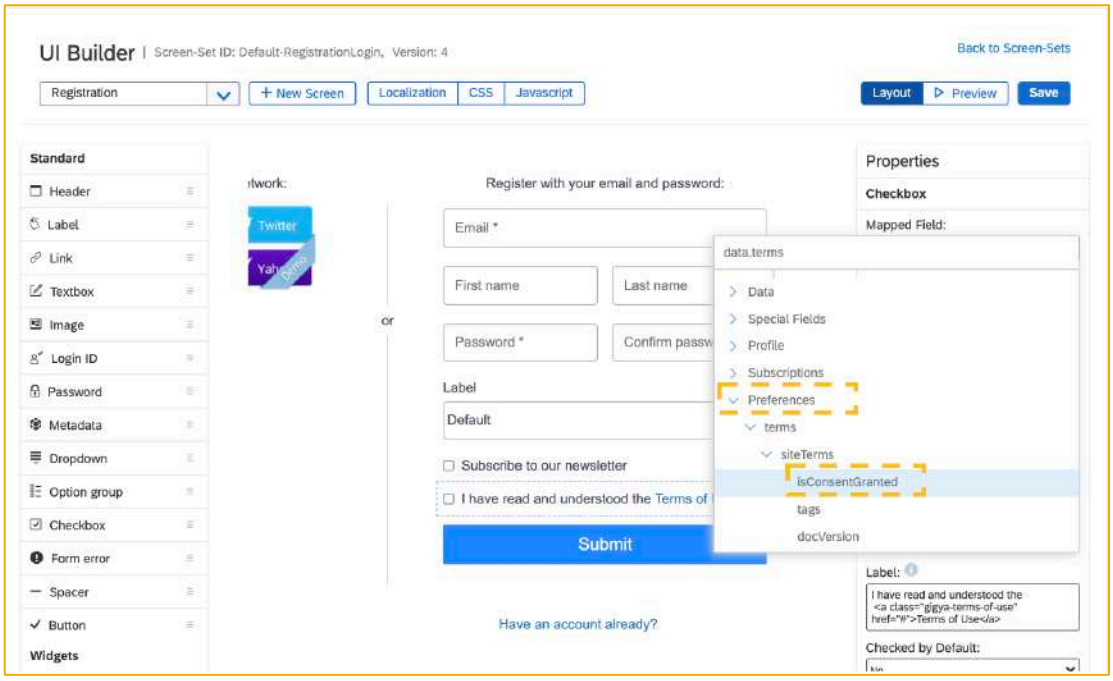
Solution

To add screen sets to a Consent Statement, follow these steps:

1. Navigate to the **User Interfacing**, **UI Builder** menu in SAP Customer Data Cloud Console
   - Open the **Default-RegistrationLogin** screen-set in the UI Builder.
   - Select the **Registration** screen from the dropdown.
   - Link the **Terms of Use** check box to the Terms Consent Statement.
   - Click in the **Mapped Field** textbox.
   - In the popup, minimize the first four top-level headings
     - Data
     - Special Fields
     - Profile
     - Subscriptions



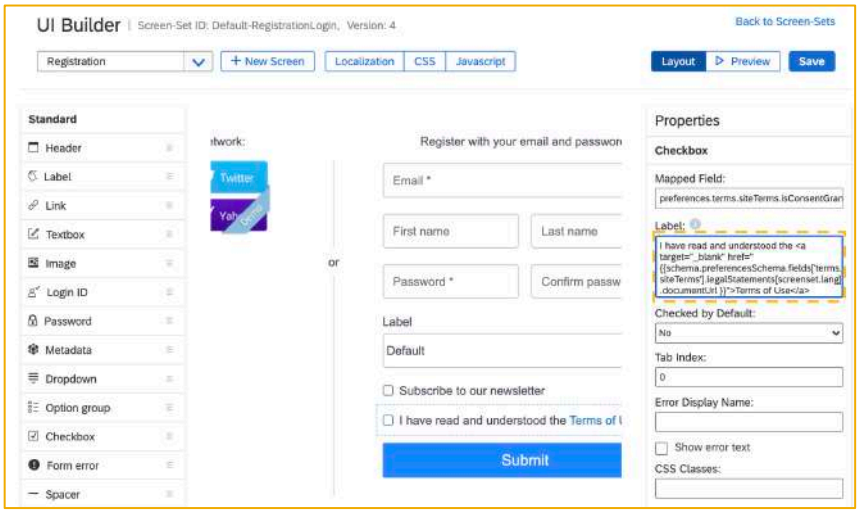
2. View the entries under **Preferences**.
   - Expand the consent statement you want.
   - Select **isConsentGranted**.



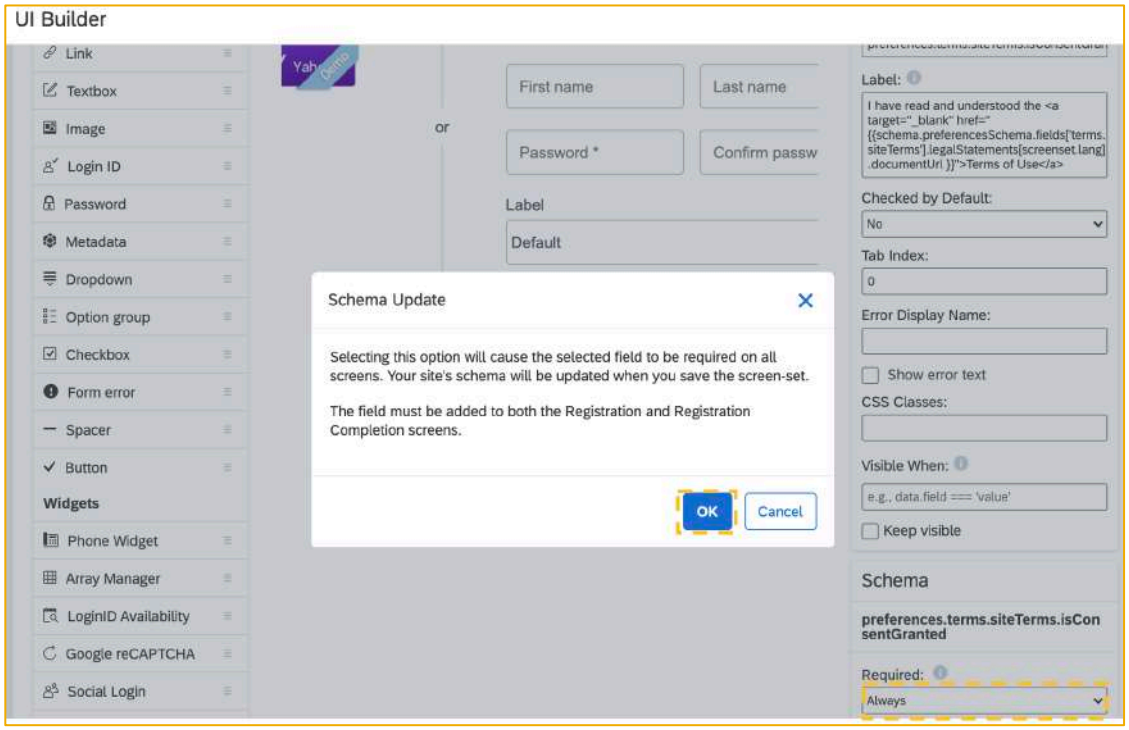
3. Update the Label to point the **Terms of Use** link to the right consent. When the customer clicks on the Terms of Use link defined on the screen-set, it should open the Terms of service document link you provided or uploaded in the Consent Statement. For that to happen, replace the following text in the Label box:

```
I have read and understood the <a target="_blank"
href="{{schema.preferencesSchema.fields['terms.siteTerms'].legalStatements[scr
eenset.lang].documentUrl }}">Terms of Use</a>
```

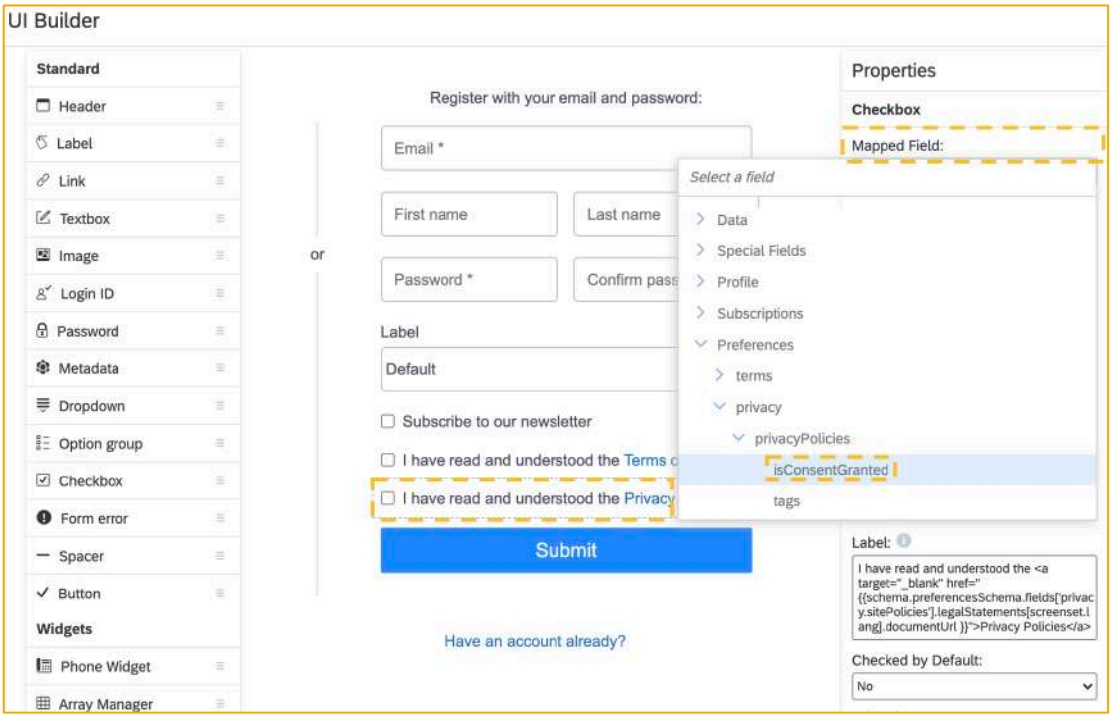
> **Note**: *siteTerms* is the ID we selected earlier for our *Terms of service* Consent.



4. Make the **Terms of Use** checkbox required (if it's not already set to that).



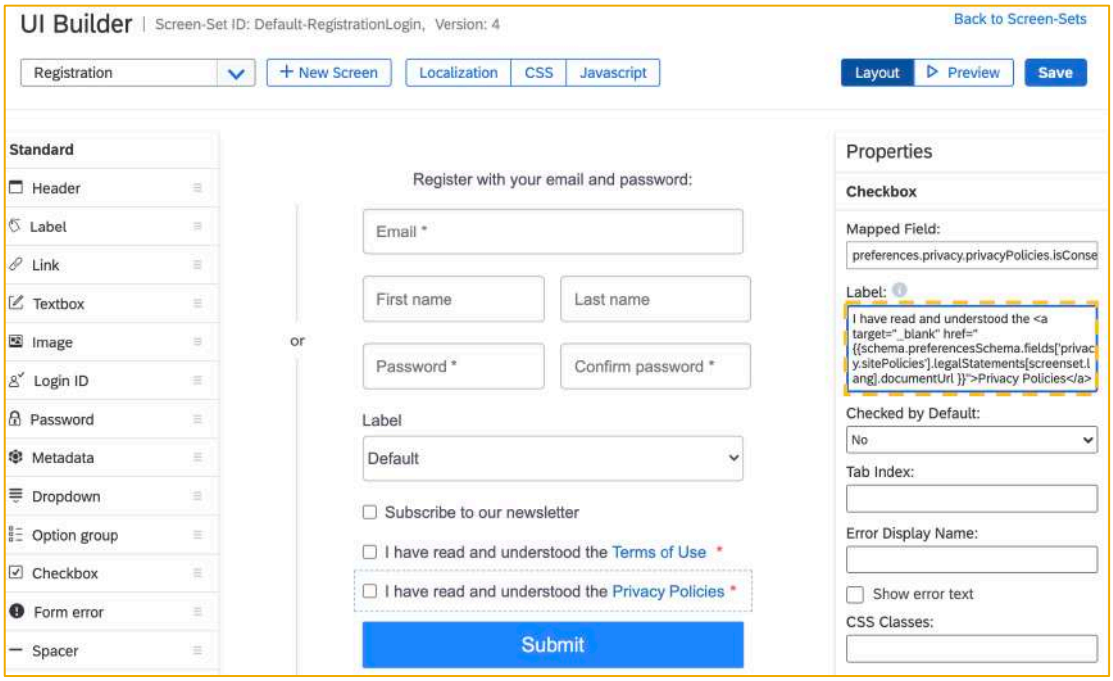
5. Add a new checkbox for **Privacy Policy** and map it to the **Policy Consent** statement.



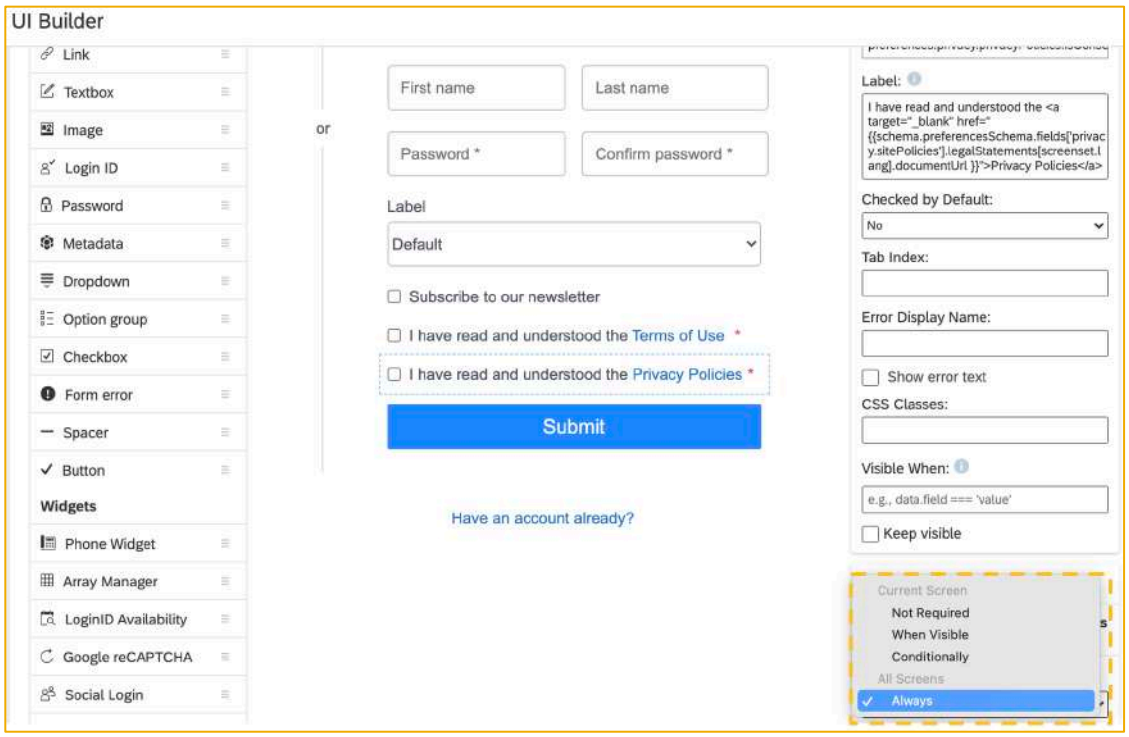
6. Update the Label with the right policy link:

```
I have read and understood the <a target="_blank"
href="{{schema.preferencesSchema.fields['privacy.sitePolicies'].legalStatements
[screenset.lang].documentUrl }}">Privacy Policies</a>
```

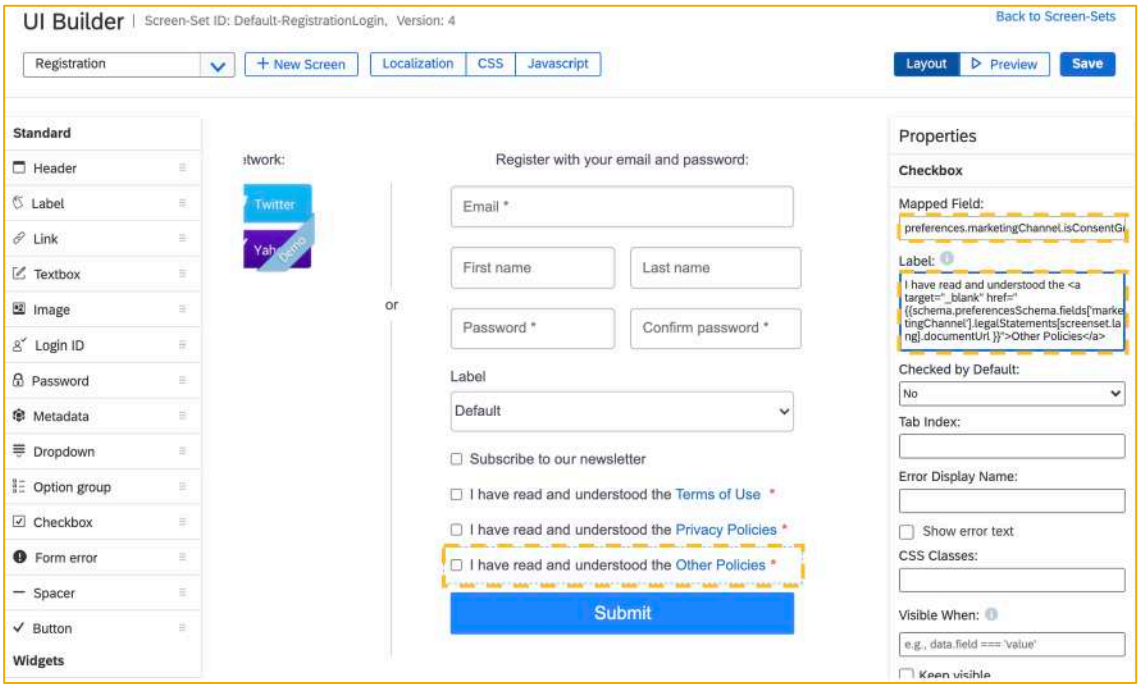
Note: *sitePolicies* is the ID we selected earlier for our *Privacy Policy* Consent.



7. Make the field required, if it isn't already.



8. Create and map a checkbox for **Other Consent Statement**.
   - Change the label.
   - Add a link to the Other Statement Consent document.
   - Keep this checkbox optional.




```
I have read and understood the <a target="_blank"
href="{{schema.preferencesSchema.fields['marketingChannel'].legalStatements[scree
nset.lang].documentUrl }}">Other Policies</a>
```

Note: siteTerms is the ID we selected earlier for our Other Consent. However, unlike for terms and Privacy Policy, the fragment you are inserting doesn't have a prefix for the "Other" type of consent. (It's just marketingChannel the ID, unlike SiteTerms and sitePolicies, which were prefixed with terms and privacy – for example, terms.siteTerms).

9. Finally, save the screen-set and close.

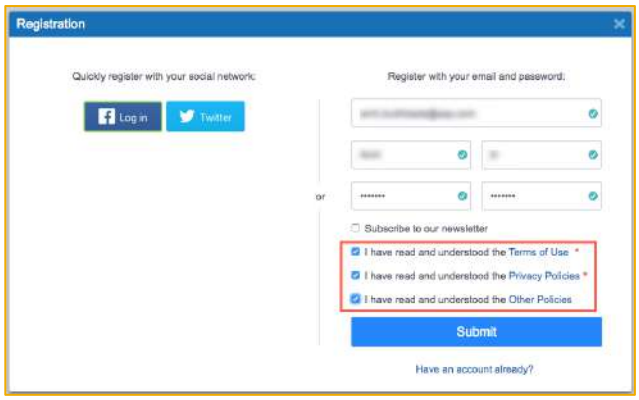
Task 3: Consent as an end user and check Consent Vault.

Once the Screen-set is created with the consent, accept the Terms of Service, Privacy Policies, and Other Policies, consent as an end user and check if consent was granted, renewed, or withdrawn in the Consent Vault.

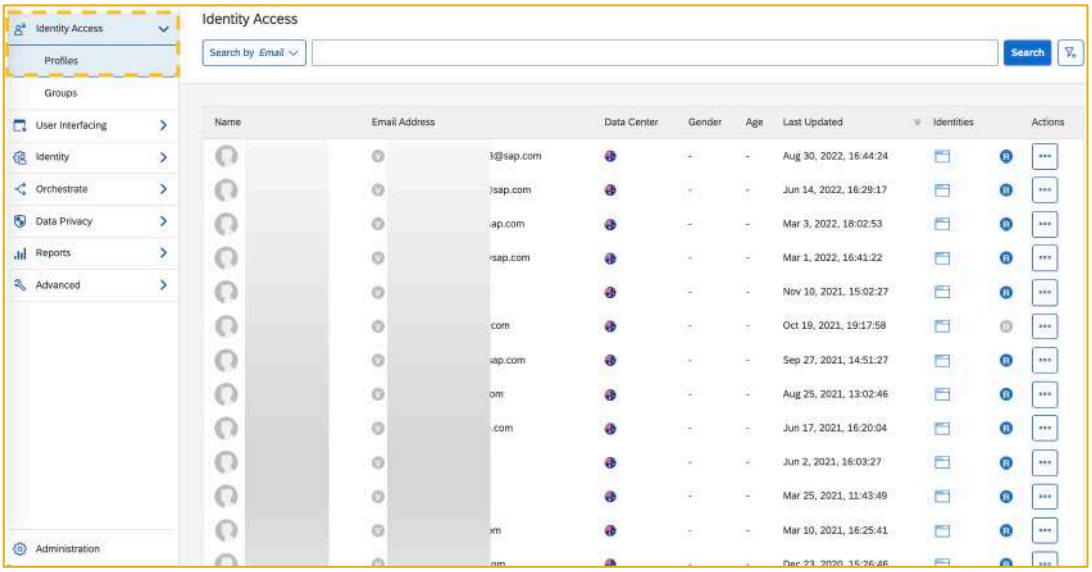
Solution

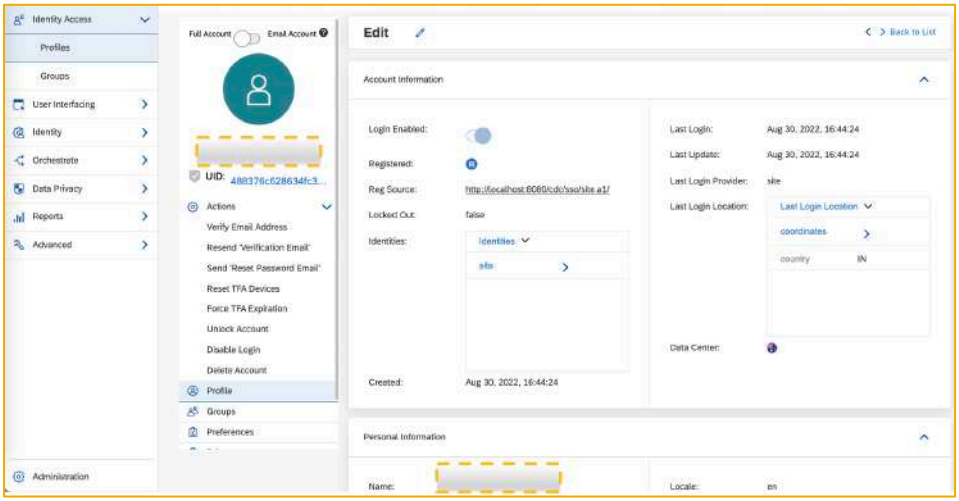
To consent as an end user and check the Consent Vault, follow these steps:

1. Register a new user with **all the three consent checkboxes selected**.

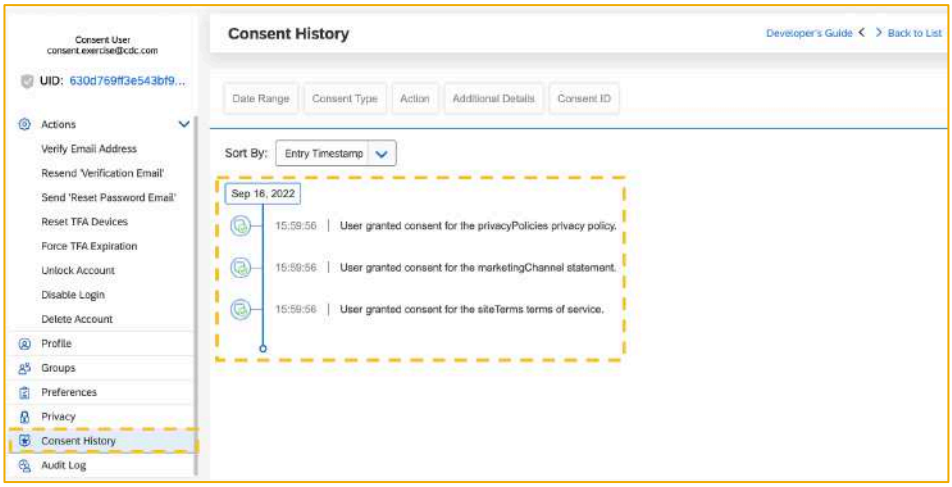


2. Navigate to **Identity Access** > **Profiles** and open the newly registered user details.

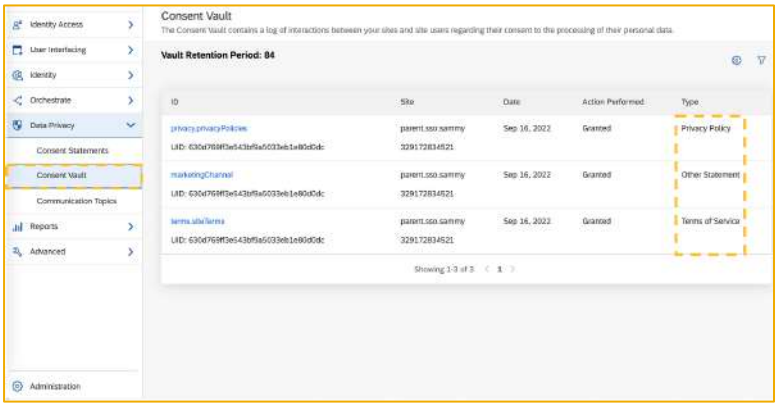




3. Click on the **Consent History** tab on the left navigation.

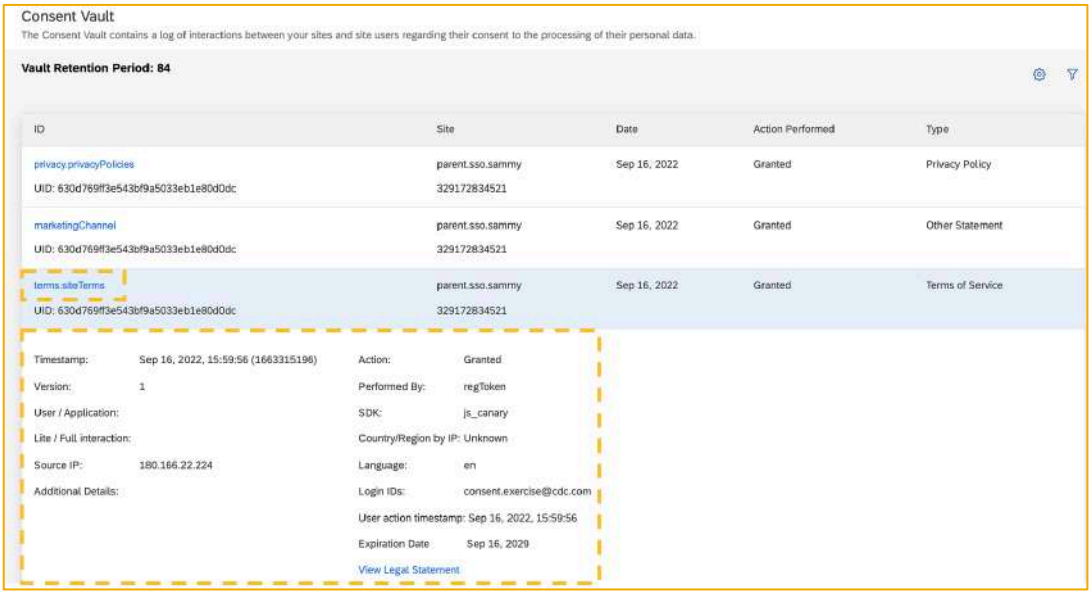


4. Navigate to **Data Privacy** > **Consent Vault** in SAP Customer Cloud Console.



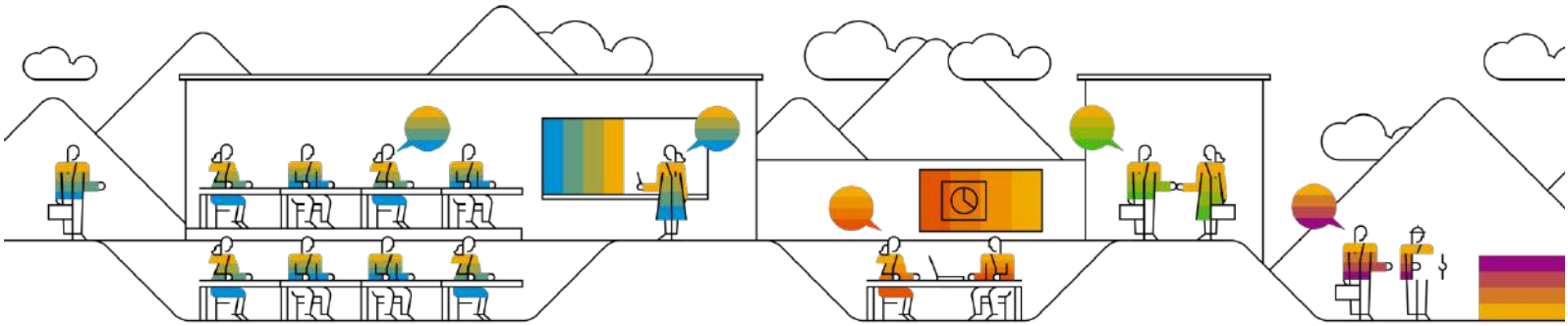
Here you can see the most recent Consent actions performed and see at a glance if consent was granted, renewed, or withdrawn.

5. Once you have found the relevant records, click on the **link in the ID column** to expand each consent.



Recap

In this exercise we created three different types of consent statements. Added the consent checkboxes in the registration screen sets, linked the checkboxes to the consent statement fields in the preferences schema. We also updated the consent links on the registration form; registered a new user with the consent selected; and checked the consent vault to confirm that.



Exercise 7: Risk Based Authentication (RBA)

Implement RBA Global Rules and RBA Account Rule-Sets

In this exercise you will create and configure RBA rules and test that they are working properly on a CDC site using the console.

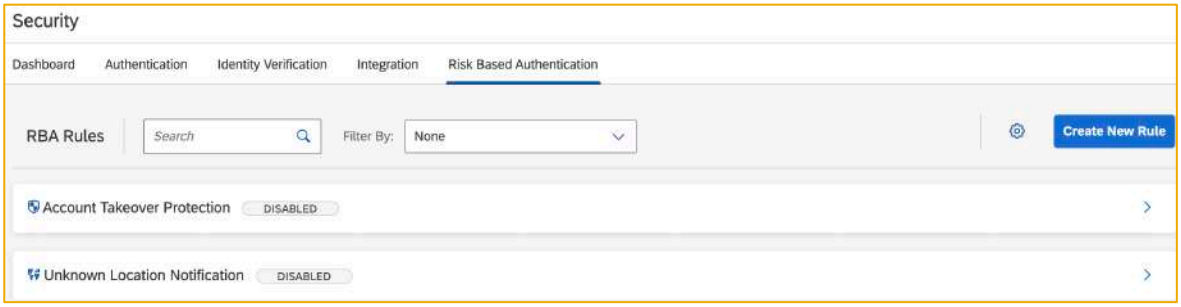
Task 1: RBA Global Rules.

Create an RBA Global Rule from the template **On multiple failed login attempts > lockout account** to lockout the user account after 5 failed login attempts.

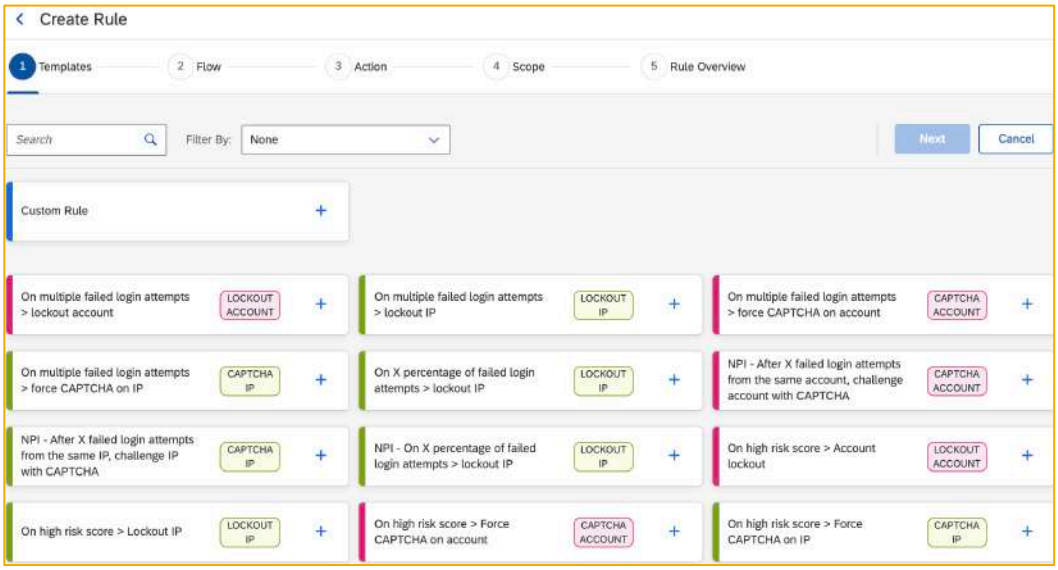
Solution:

To create an RBA Global Rule, follow these steps:

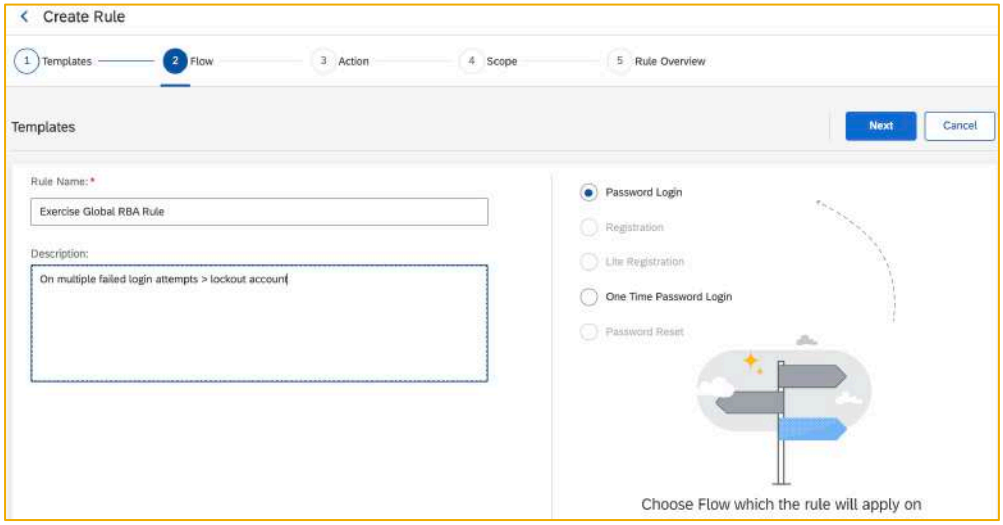
1. Log into CDC's console, select **Identity, Security** from the navigation menu of your CDC site**.**
2. Select the **Risk Based Authentication** tab.



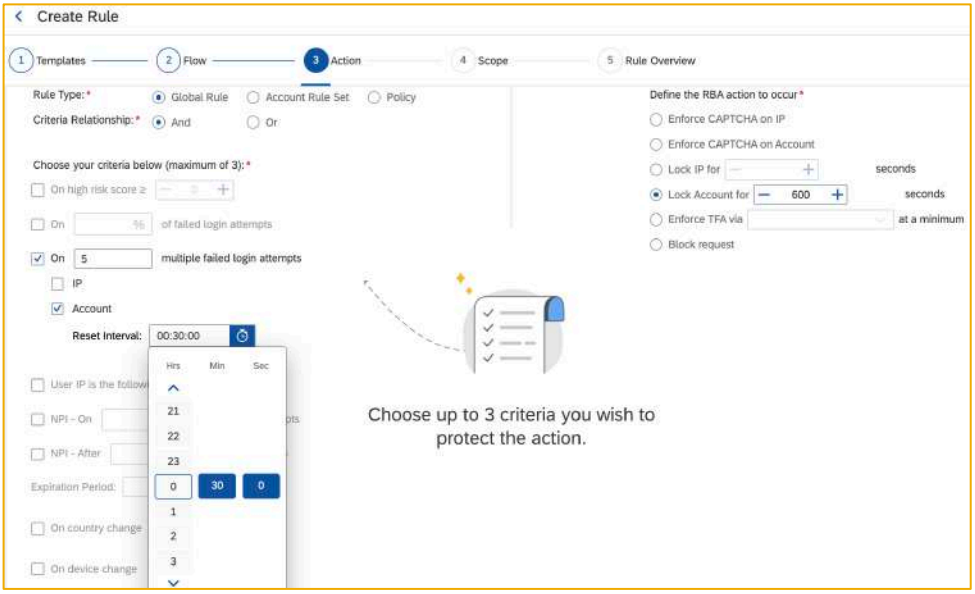
3. Click on the **Create New Rule** button on the right side.
4. On the first step, **Templates**
   - Select from the list of templates the **On multiple failed login attempts > lockout account** template (usually the first one).
   - Click **Next**.



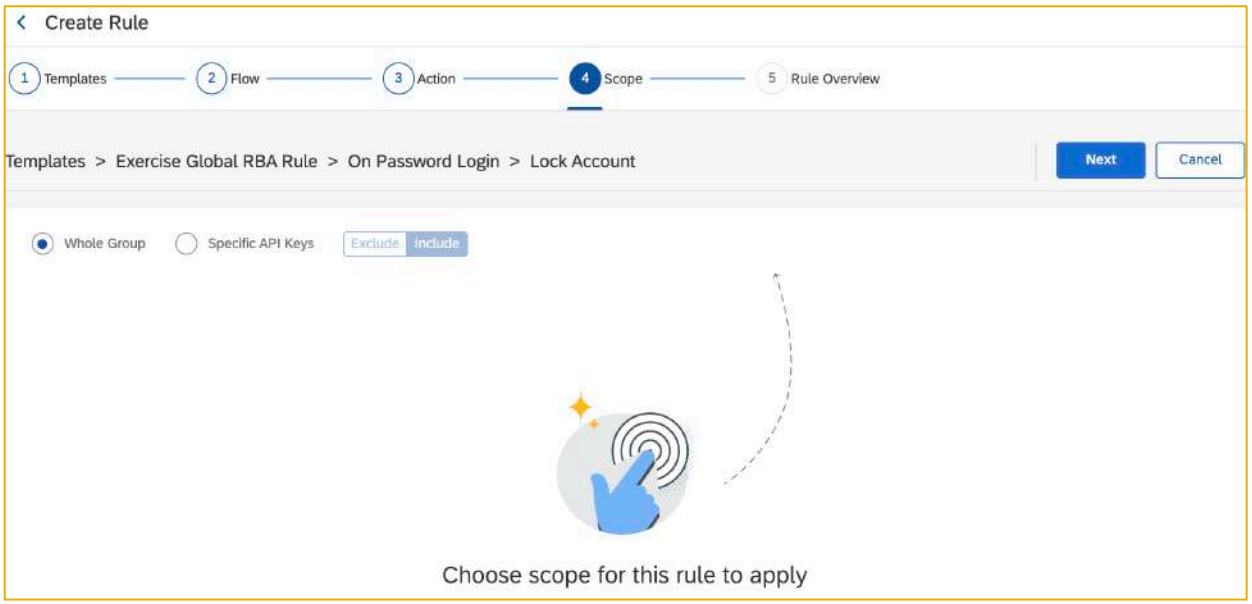
5. On the **Flow** step
   - Enter a **name and description** for the new rule.
   - Select the **CDC flow** to apply the new rule.



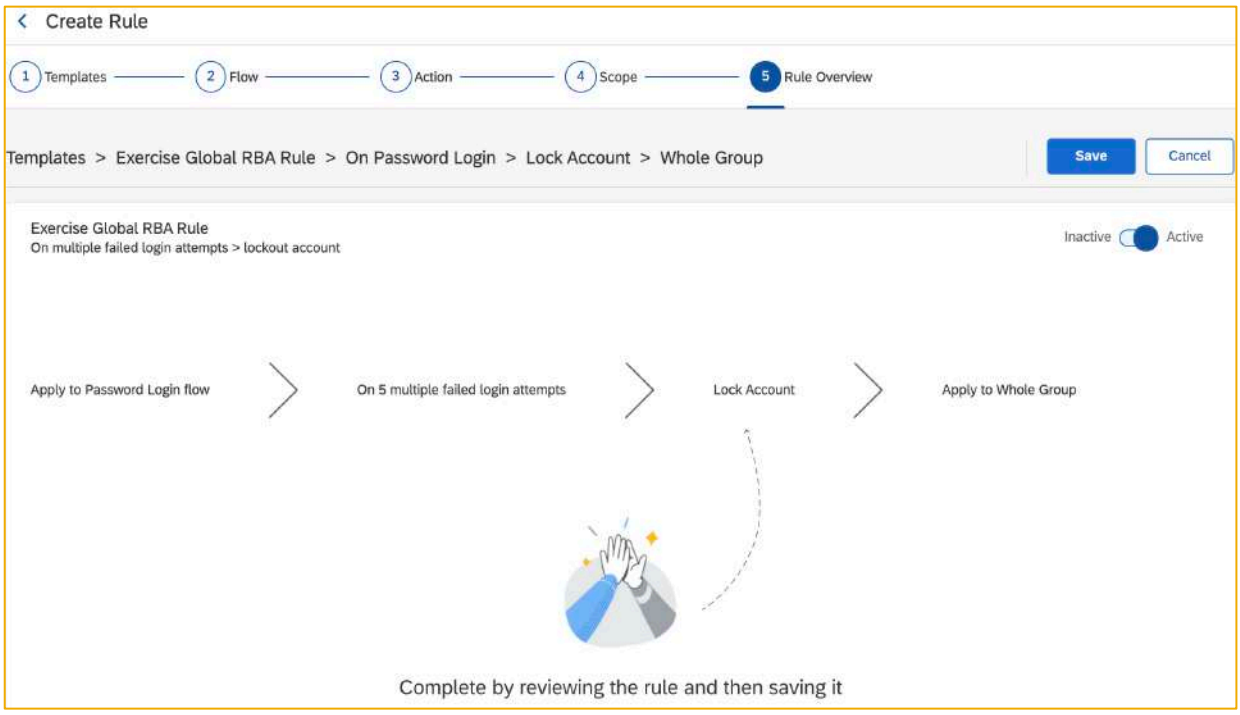
6. On the **Action** step, enter the following:
   - Rule Type: select **Global Rule**.
   - Criteria Relationship: select **And**.
   - The 'On 10 multiple failed login attempts' criteria is pre-selected from the rule template. Change the threshold number from 10 to **5**.
   - Account scope checkbox: the Reset Interval is pre-populated with 1 hour from the rule template. Click on the **clock** icon to change the amount of time, for example 30 minutes. The configuration means: when 5 (threshold) failed logins (type) for the same account (scope) occur within 30 minutes (resetInterval).
   - Define the RBA action to occur: Lock Account RBA action is also pre-selected from the rule template. You can change the default lockout period from 600 seconds to a different period to meet your requirements (optional). The RBA action means: lockout (type) the account (scope) for 10 minutes (600 seconds - duration).
   - Click **Next**.



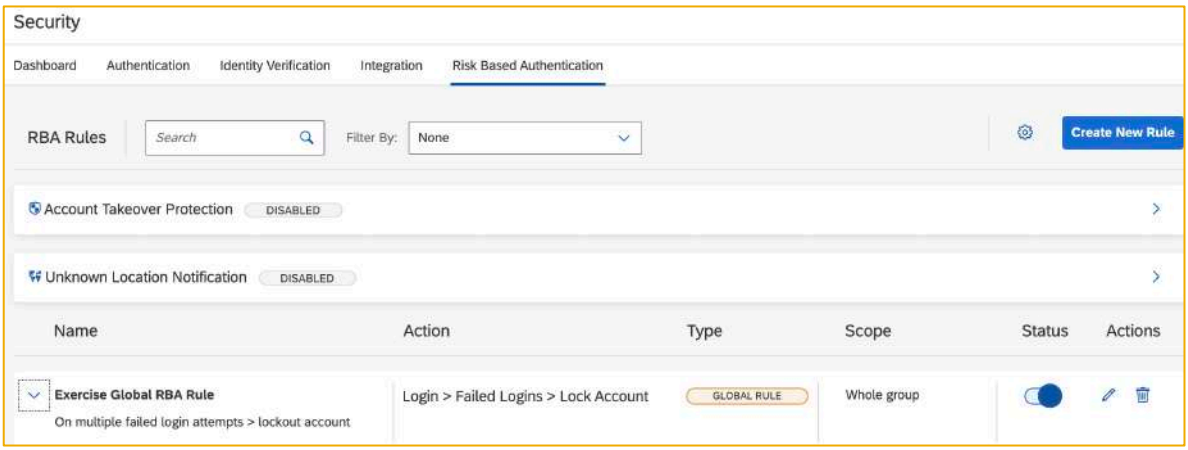
7. On the Scope step
   - Use the default **Whole Group** scope.
   - Click **Next**.



8. On the last step, Rule Overview, review the rule configurations. If something is wrong or you want to change something, select the steps buttons to go back to the previous steps. Otherwise, keep the **Active** toggle on.
   - Click **Save** to finish your RBA rule creation.



Your new RBA rule should be listed on the Risk Based Authentication page.

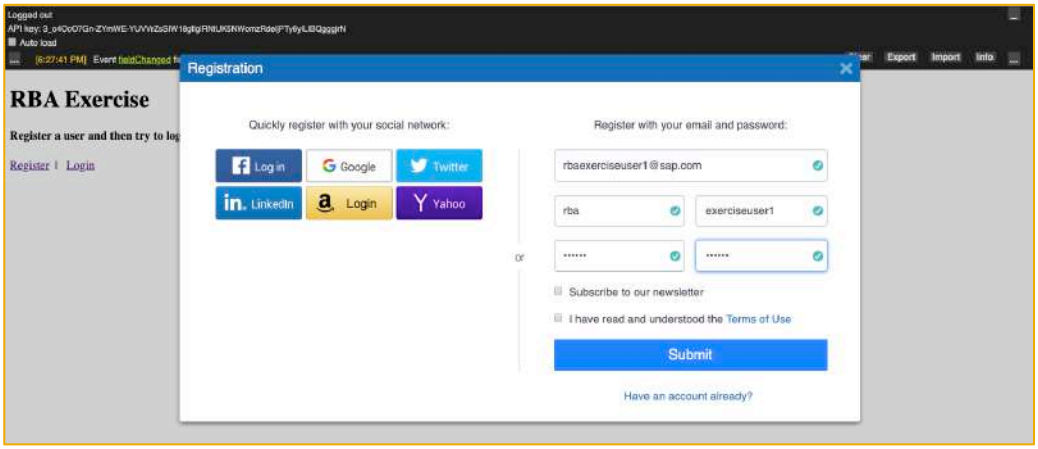


9. To test the new RBA Global Rule, create a **new HTML** page with the following code or reuse the registration / login links from any previous exercise:

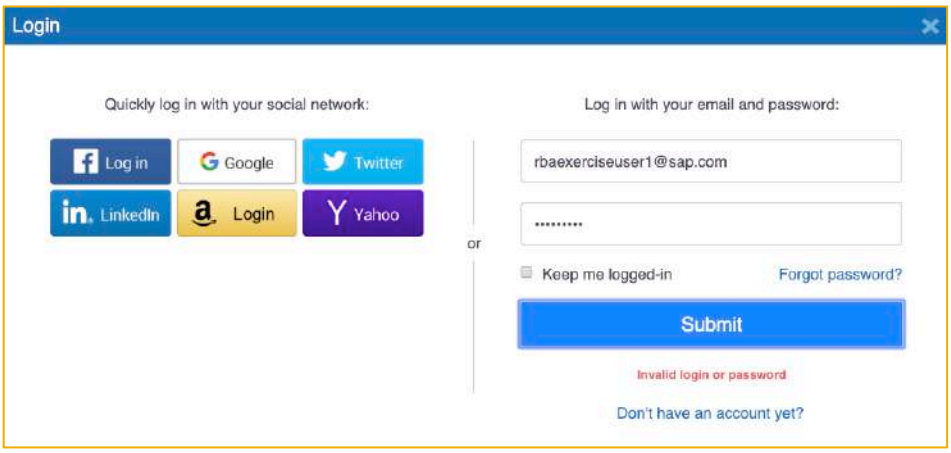
```
      <html>
  <head><title>RBA Exercise</title>
    <SCRIPT type="text/javascript" lang="javascript"
            src="http://cdn.gigya.com/js/gigya.js?apikey=YOUR-API-KEY">
    </SCRIPT>
  </head>
  <body><h1>RBA Exercise</h1>
  <h4>Register a user and then try to login with a wrong password 5 times</h4>
      <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin', startScreen:'gigya-register-
screen'});">Register</a>  |  
    <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin'});">Login</a>
    <span id="logout" hidden>  |  <a href="#"
onclick="logout();">Logout</a></span>
    <script type="text/javascript" lang="javascript">
        function logout() {
              gigya.accounts.logout({callback:function (response) {
                if (response.errorCode == 0) {
                  document.getElementById("logout").hidden=true; }
                else {
                  alert('Error:' + response.errorMessage);
                }
              }});
        }
```

```
        gigya.accounts.getAccountInfo({callback:function (response) {
            if (response.errorCode == 0) {
              document.getElementById("logout").hidden=false;
            }
        }});
       gigya.socialize.addEventHandlers({
          onLogin: function () {
            document.getElementById("logout").hidden=false;
        }
      });
    </script>
  </body>
</html>
```

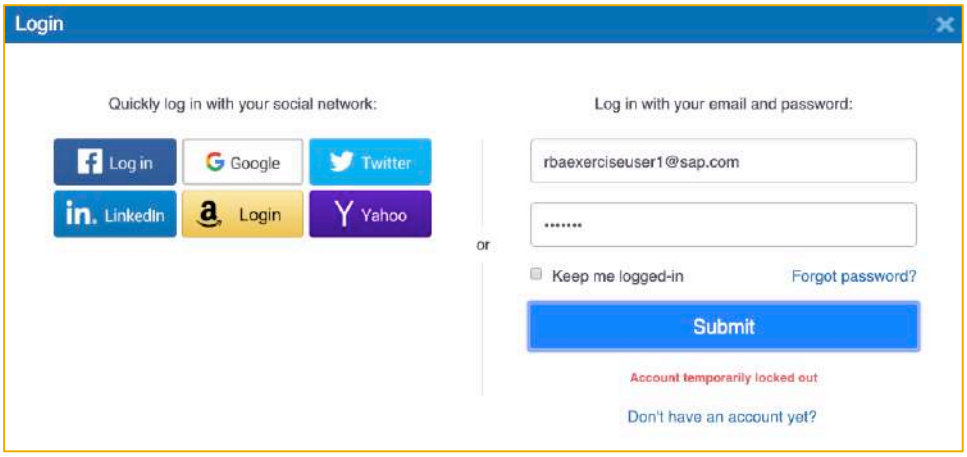
10. Save the **HTML page** in the root folder of your local web server and navigate to it.
11. Then, click on the **Register** link to create a new user account.



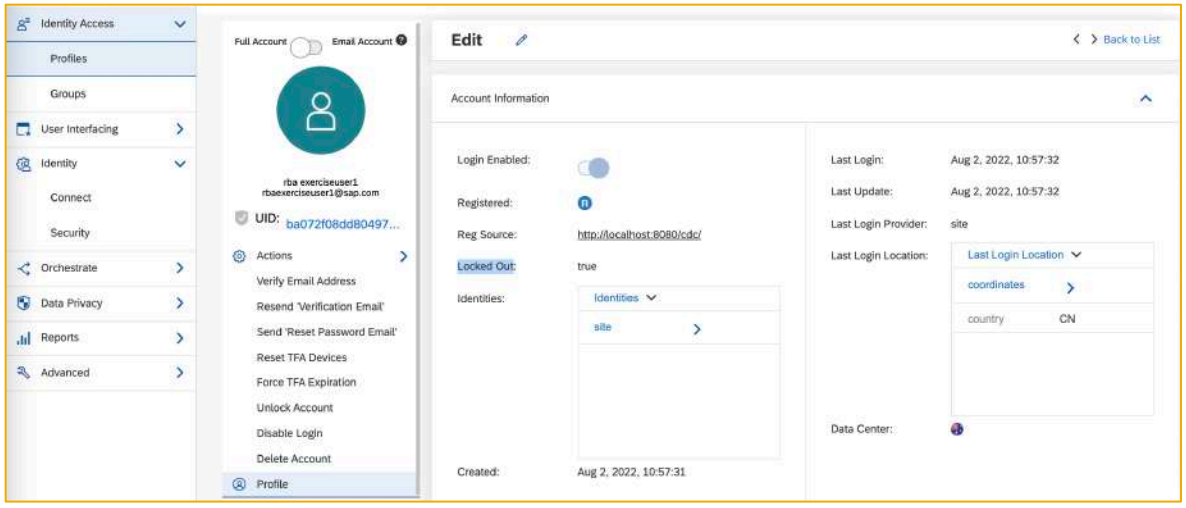
12. Log the user out and try to log in again using a wrong password.



13. Try to login 4 more times. The account then gets temporarily locked.



14. Go to the console. Inside Identity Access look for the user and open its record. Notice the **Lock Out** Boolean flag.



You could also unlock the account using the actions menu; however, this account will be unlocked after 10 minutes.

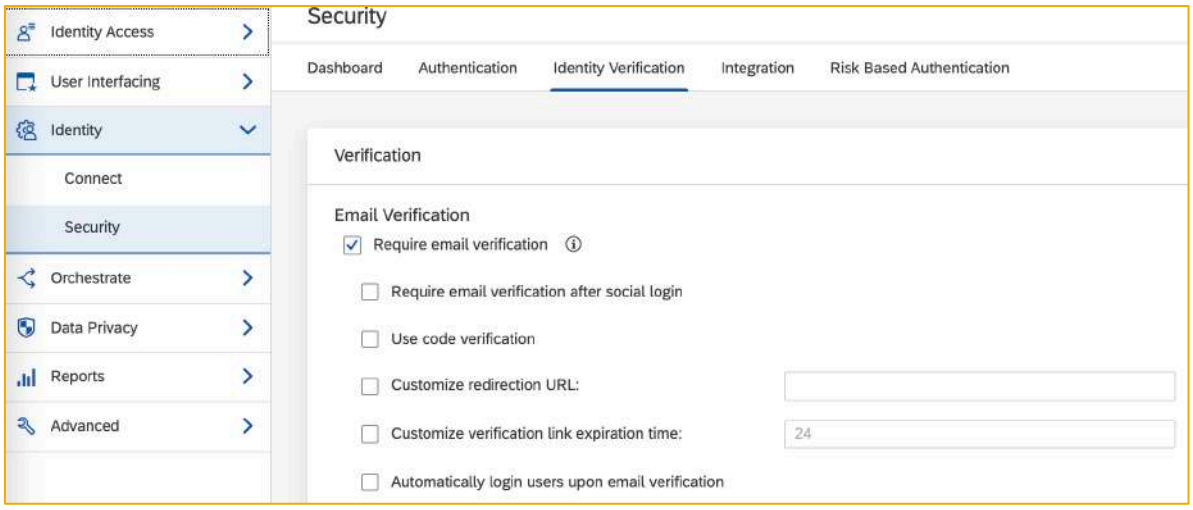
Task 2: RBA Account Rule-Sets

Create an RBA Account Rule-Set from the template **On device change > force verification of auth level 10 or higher** to implement email verification of the user account.

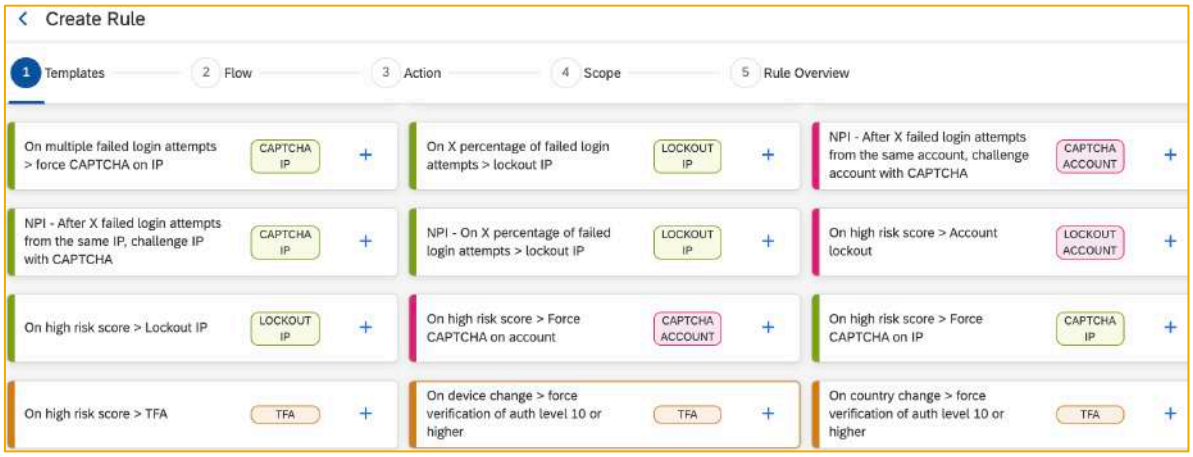
Solution

To create an RBA Account Rule-Set, follow these steps:

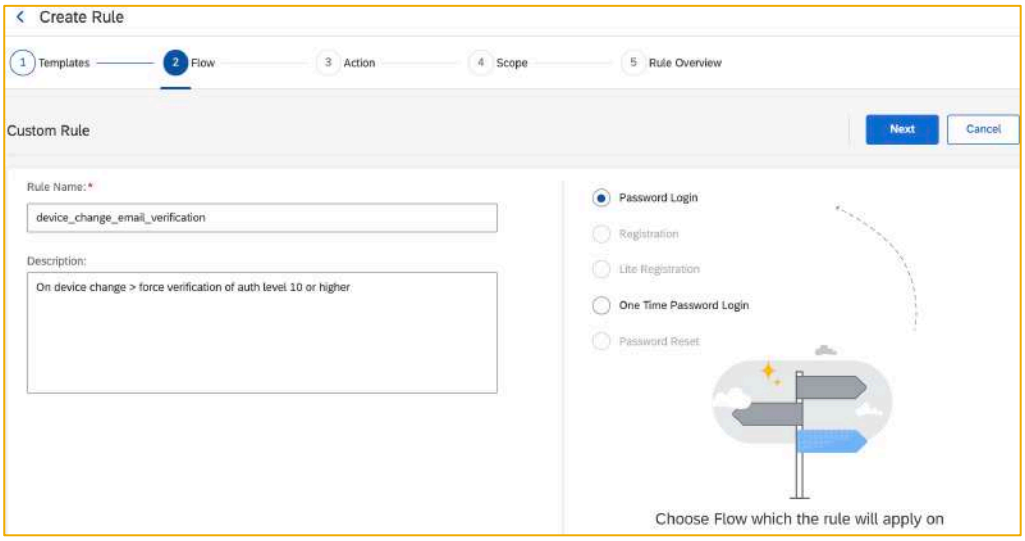
1. Log into CDC's console and select **Identity > Security** from the navigation menu of your CDC site.
2. Select the **Identity Verification** tab.
   - Check the box **Require email verification** under Email Verification (This setting is required for email OTP later).
   - Click **Save** at the bottom of the page.



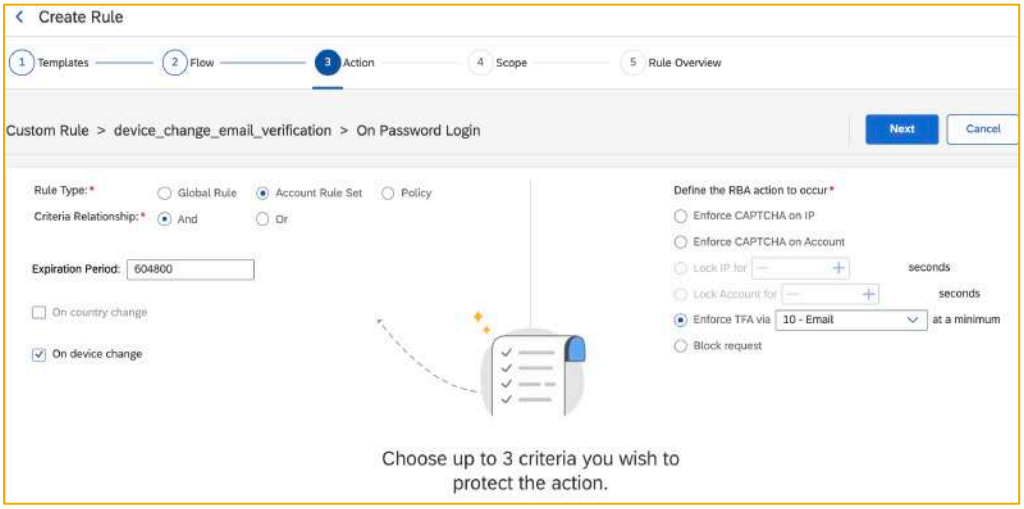
3. Click on the **Risk Based Authentication** tab.
4. Click on **Create New Rule** on the right side.
5. On the first step, **Templates**
   - Select from the list of templates the **On device change > force verification of auth level 10 or higher.**
   - Click **Next**.



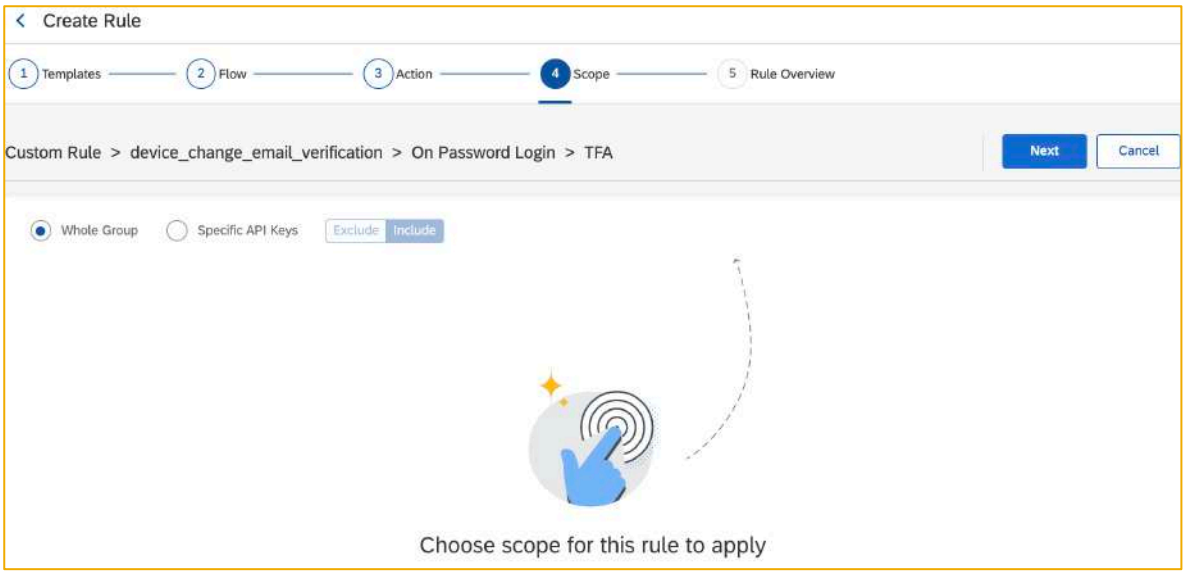
6. On the Flow step
   - Enter a name and description for the new rule; the default name is **device_change_email_verification.**
   - On the right side, select the CDC flow on which to apply the new rule.
   - Click **Next**.



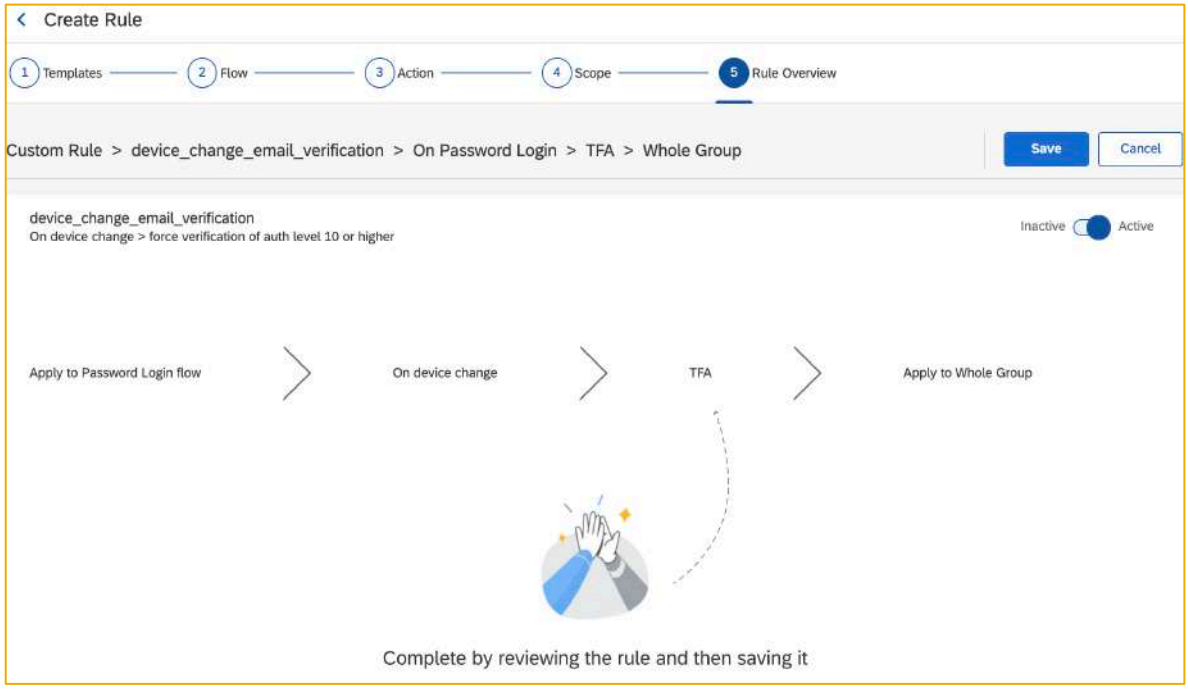
7. On the Action step
   - Everything is already pre-selected from the rule template, eg. Rule Type, Criteria Relationship and Expiration Period etc.
   - You can optionally change the Expiration Period from the default 2592000 seconds (30 days) to something else, such as a week, which is 604800 in seconds.
   - On the right side, the RBA action is also pre-selected to auth level 10 to Enforce TFA via email. The condition means: when the physical device or unique browser (device) used to access the site changes, or every 7 days (604800 secs - expirationPeriod). The action part means forcing the user to pass to the Two-Factor Authentication (type) mechanism associated with authLevel 10 or higher.
   - Click **Next**.



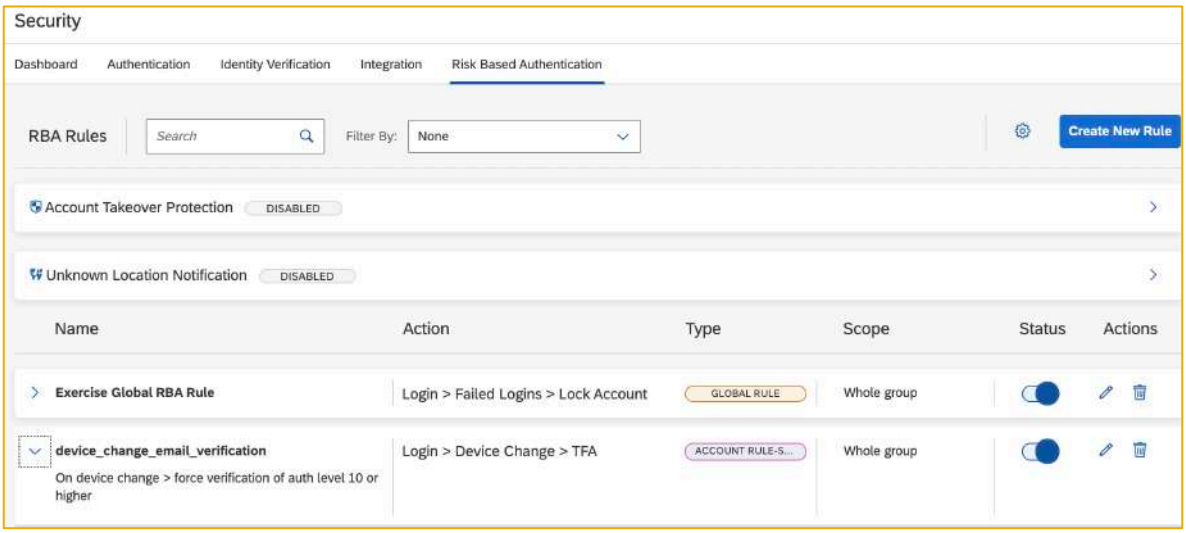
8. On the Scope step
   - Use the default **Whole Group** scope.
   - Click **Next**.



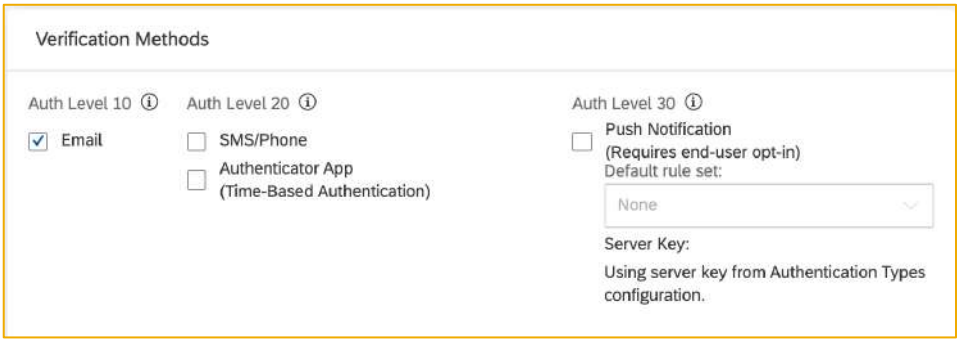
9. On the last step, **Rule Overview,** review the configurations. If something is wrong or you want to change something, select the steps buttons to go back to the previous steps. Otherwise, keep the **Active toggle on**.
   - Click **Save** to finish your RBA rule creation.



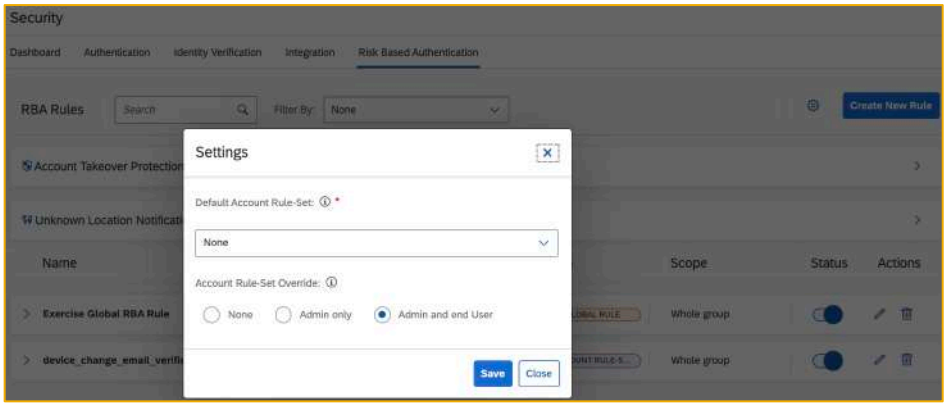
Your new RBA rule should be listed on the Risk Based Authentication page.



10. Select the **Authentication** tab.
    - Select the **Email** option under Verification Methods at the bottom of the page to select the Verification Method associated for Auth Level 10, in this case Email OTP (One-time password).



11. Select the Risk **Based Authentication** tab.
    - Click the gear icon on the left side of the Create New Rule button to bring up the account rule-set settings popup.
    - Enable the Account Rule-Set Override for the **Admin and End User** to allow the end user to enable the usage of a particular Account Rule-Set (the one we have just created).



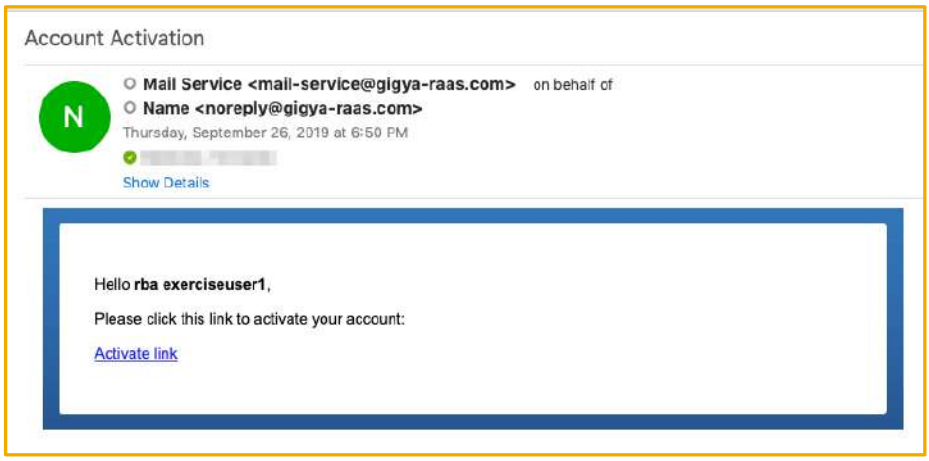
12. To test the new RBA Account Rule, create a **new HTML page** with the following code or reuse the registration / login links from any previous exercise:

```
      <html>
  <head><title>RBA Exercise</title>
    <SCRIPT type="text/javascript" lang="javascript"
            src="http://cdn.gigya.com/js/gigya.js?apikey=YOUR-API-KEY">
    </SCRIPT>
  </head>
  <body><h1>RBA Exercise</h1>
  <h4>Register a user and then try to login with a different browser</h4>
      <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin', startScreen:'gigya-register-
screen'});">Register</a>  |  
    <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin'});">Login</a>
    <span id="logout" hidden>  |  <a href="#"
onclick="logout();">Logout</a></span>
    <script type="text/javascript" lang="javascript">
        function logout() {
              gigya.accounts.logout({callback:function (response) {
                if (response.errorCode == 0) {
                  document.getElementById("logout").hidden=true; }
                else { alert('Error:' + response.errorMessage);
               }
             }});
        }
        gigya.accounts.getAccountInfo({callback:function (response) {
            if (response.errorCode == 0) {
              document.getElementById("logout").hidden=false;
            }
        }});
       gigya.socialize.addEventHandlers({
          onLogin: function () {
            document.getElementById("logout").hidden=false;
        }
      });
    </script>
  </body>
```

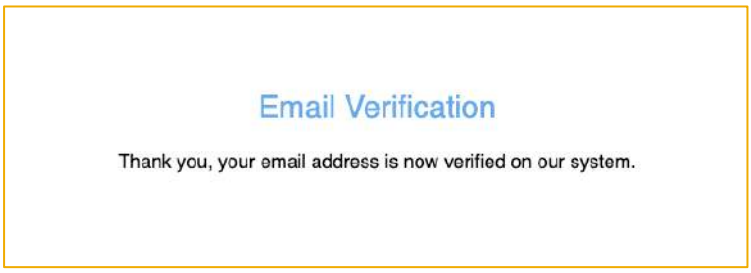
```
</html>
```

13. **Save** the HTML page in the root folder of your local web server and navigate to it.
14. Then click on the **Register** link to create a new user account with a **valid email address**.

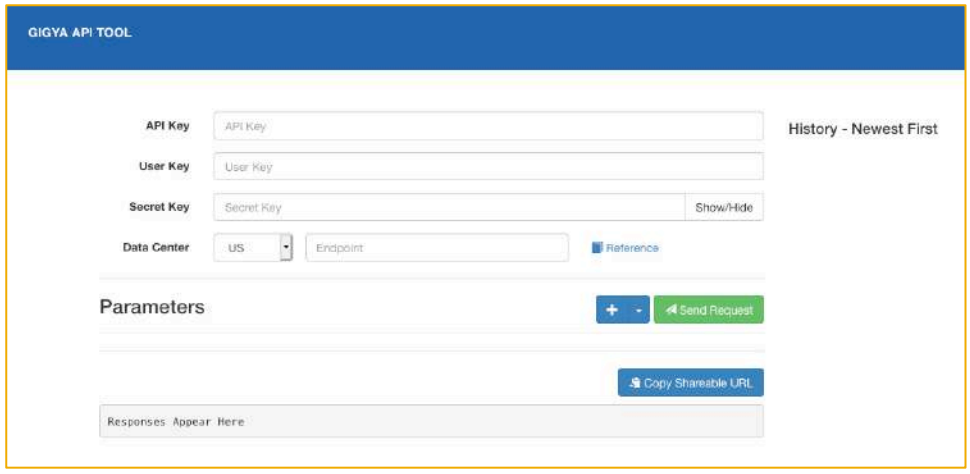
    Wait few seconds to receive the activation email and click on the Activate link.



Now the email has been verified.

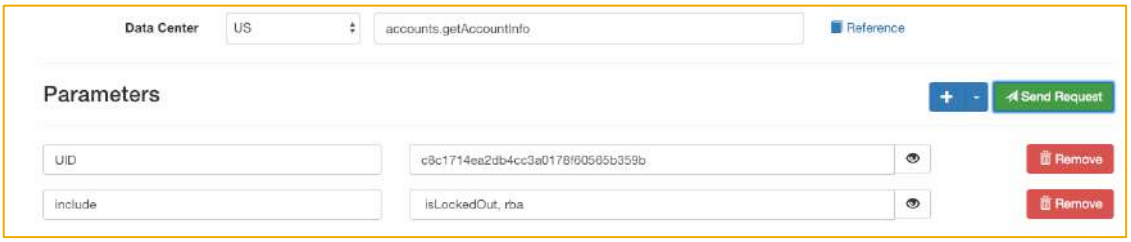


15. In this part of the exercise, we are going to use REST API to assign the previously created rule-set to our registered users. Go to https://tools.gigya-cs.com/api/ .



16. Enter the data related to the API Key, User Key, and Secret Key for your exercise site.

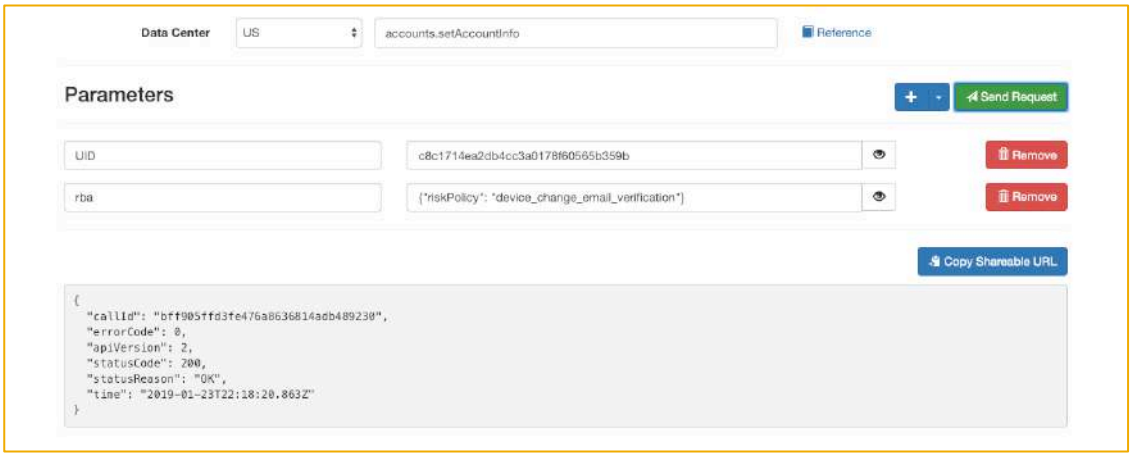
17. Check that initially there is no custom RBA policy for this user. Use the **accounts.getAccountInfo** REST API endpoint with the recently registered user's ***UID*** value and ***include*** parameter with the options: *isLockedOut, rba*.



18. This request will return the initial RBA configuration. (There is no active policy and the **riskPolicyLocked** contains a false value which allows the user to change it).

```
"rbaPolicy": {
  "riskPolicyLocked": false
},
```

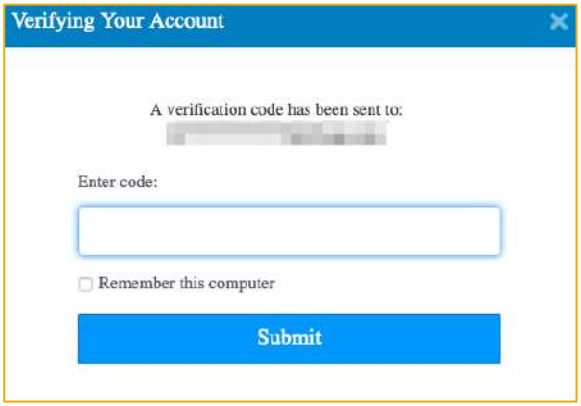
19. If you now invoke **accounts.setAccountInfo** with the same **UID** and the **rba** parameter set to ***{'riskPolicy': 'device_change_email_verification'}*** you'll be assigning the custom risk policy to this particular user instead of using the Default Account Rule-Set.



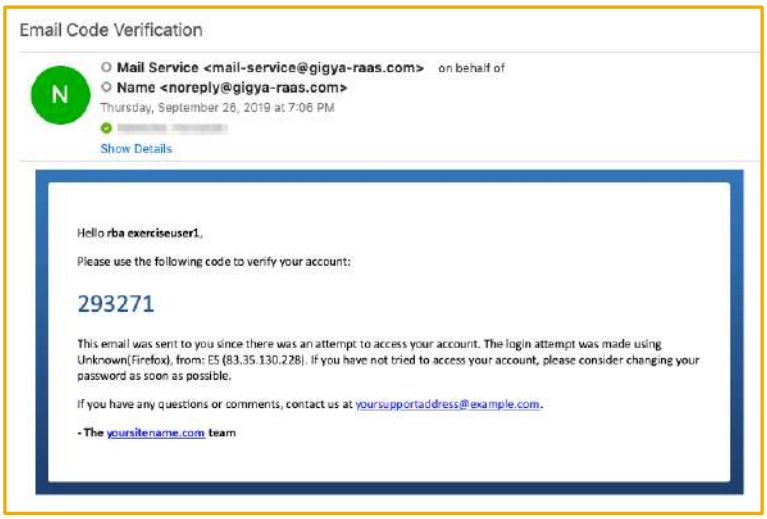
20. Request **accounts.getAccountInfo** again with the UID and include parameters adjusted to show that the Account rule-set has been assigned for this user:

```
"rbaPolicy": {
  "riskPolicy": "device_change_email_verification",
  "riskPolicyLocked": false
},
```

21. (Using another web browser / device) go back to the HTML page you've created. Click on **Login** and enter a valid email and password **using another web browser**. The **Verifying Your Account** screen is shown.



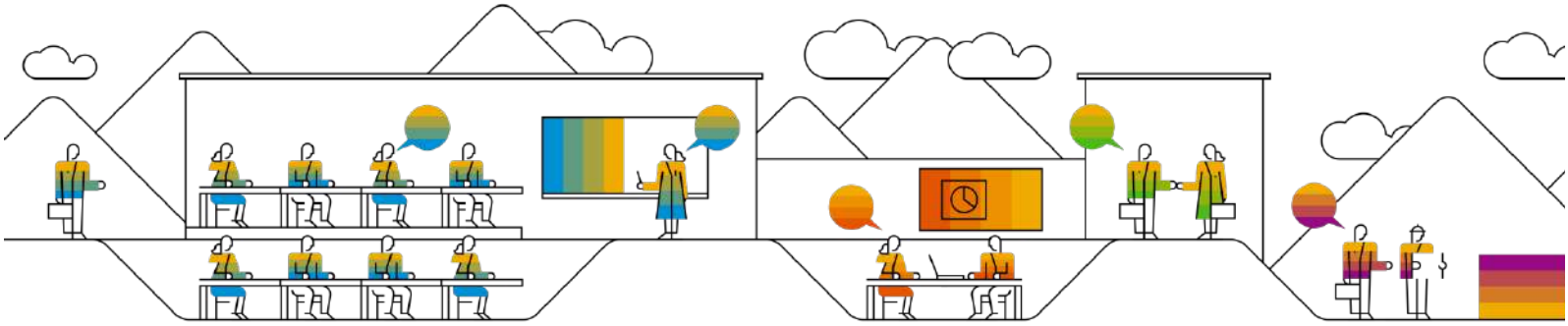
22. Check your email inbox. There will be a new message with the subject *Email Code Verification* and a number (OTP).



23. Enter the **OTP** code (293271) into the screen and complete the login process. You should be able to see the Logout link on your web page.

Recap

In this exercise, you learned how to create Global RBA rules and RBA Account Rule-Sets, as well as how to assign a particular Account Rule-Set to a specific customer account using REST APIs.



Exercise 8: Extensibility – JavaScript Parameters

Customize the Customer Registration Process

In this exercise you will use one of the CDC's extensibility mechanisms JavaScript Parameters. You will learn how to customize the customer registration process with custom codes running in client-side browsers.

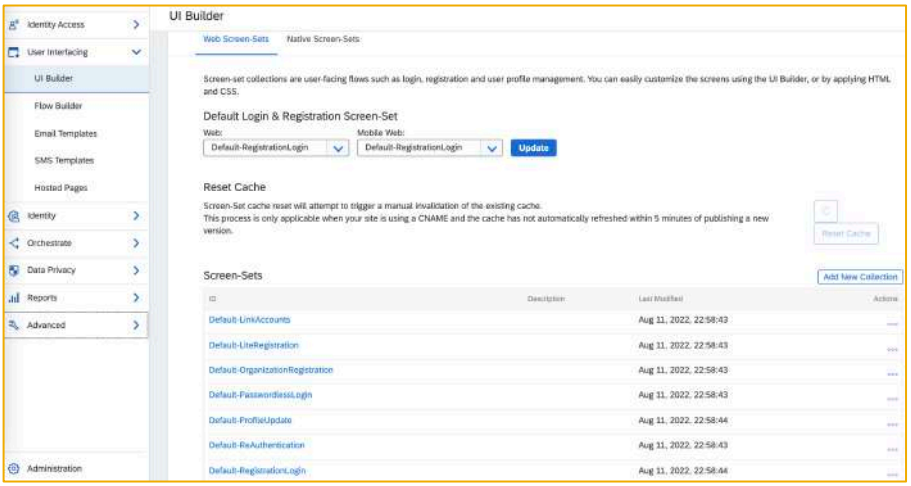
Task 1: Javascript Parameter Creation.

Customize the Registration screen with custom client logic which will prevent the customer to register with his first name not formed by letters (only vowels and consonants will be allowed).

Solution

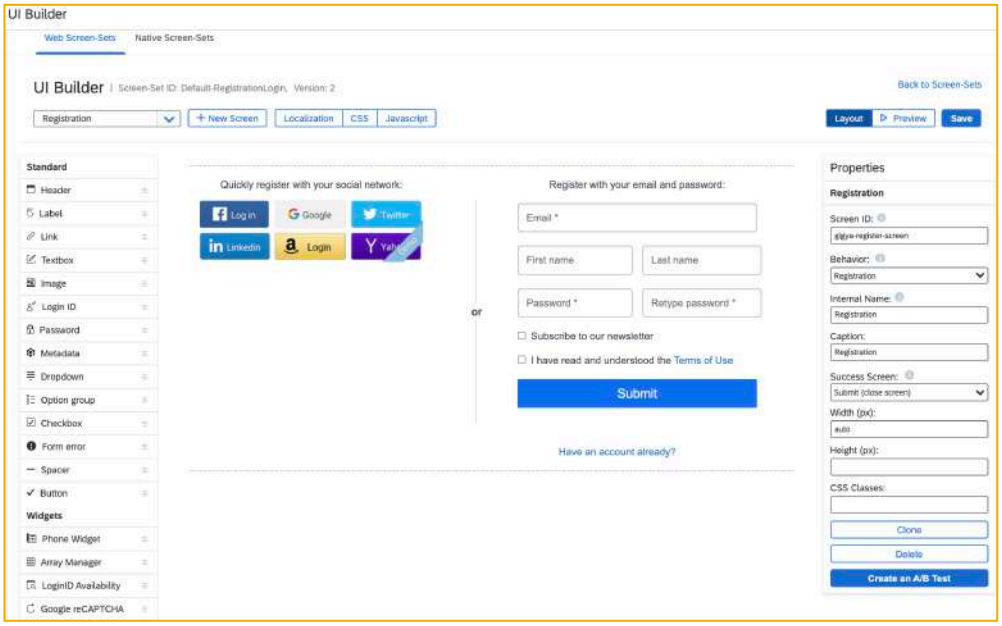
To customize the customer registration screen, follow these steps:

1. Login into **CDC's console** and select **User Interfacing** > **UI Builder** from the navigation menu of your site.
   - Select the **Web Screen-Sets** tab, which is selected by default.

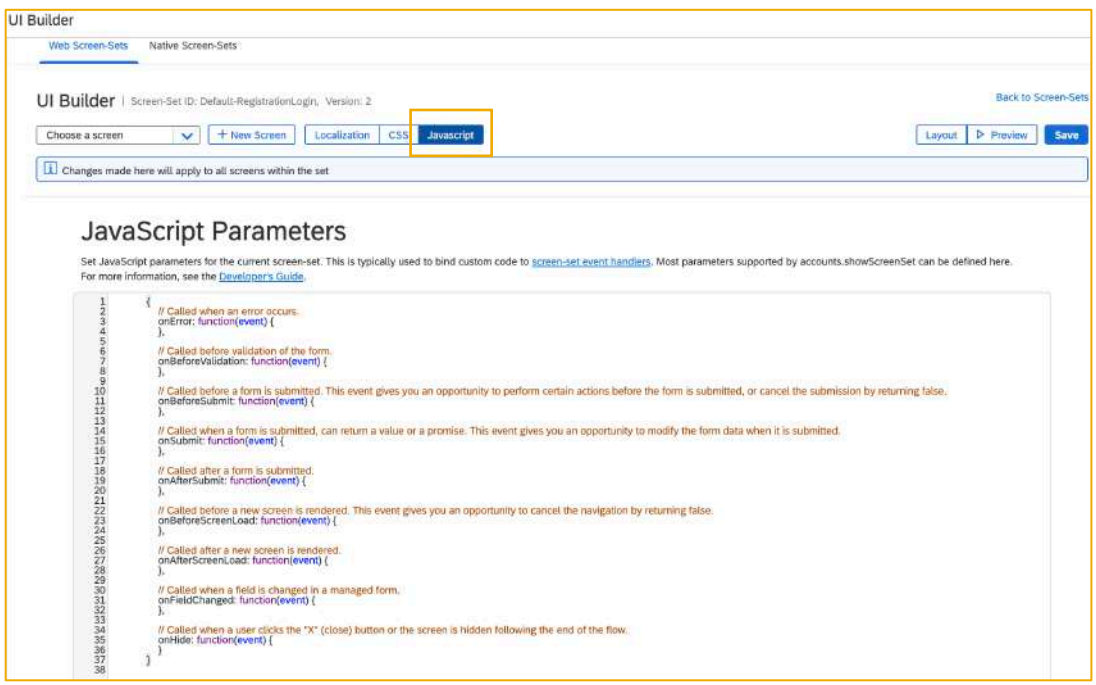


2. Select the **Default-RegistrationLogin** screen-set to Enter in the UI Builder and select the R**egistration** screen from the dropdown.

   Your screen should look like this:



3. Click on the **JavaScript Parameters** menu.



4. Edit the **onBeforeValidation** function parameter and enter the following code between its curly braces:

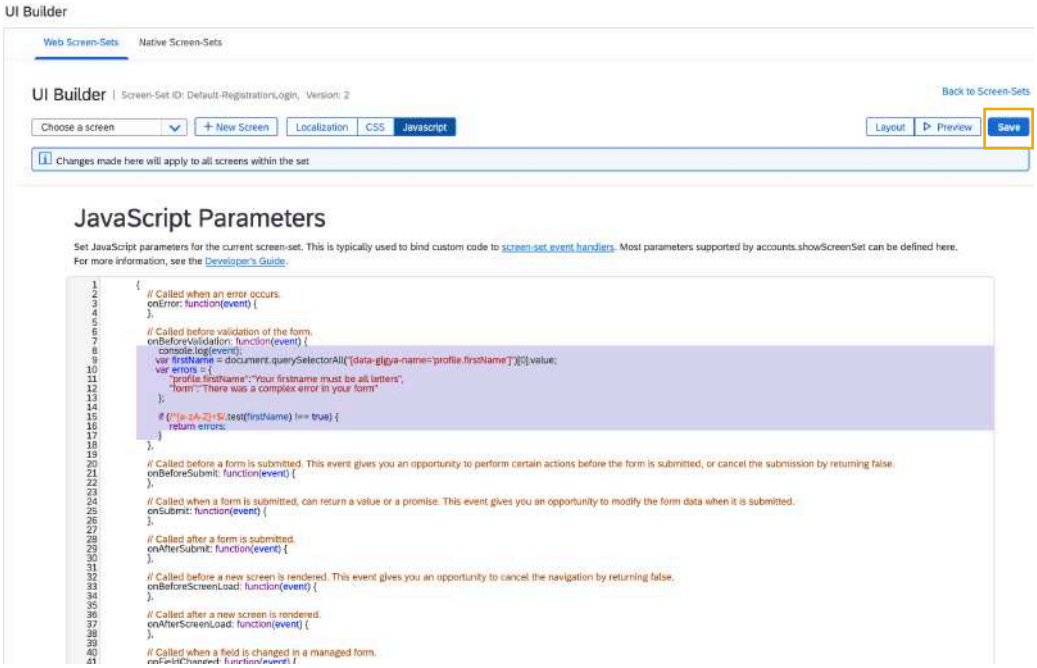
```
            console.log(event);
        var firstName = document.querySelectorAll("[data-gigya-
name='profile.firstName']")[0].value;
        var errors = {
            "profile.firstName":"Your firstname must be all letters",
            "form":"There was a complex error in your form"
```

```
        };

        if (/^[a-zA-Z]+$/.test(firstName) !== true) {
            return errors;
        }
```

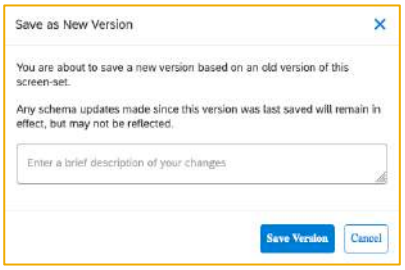
The code goes through the following steps:

1. Log the **event object** in the browser console.
2. Select the **value of the UI control** that is mapped (data-gigya-name) to the profile.firstName schema field.
3. It creates two different kinds of error messages (field error and form error) under an errors object variable.
4. Later it checks against the regular expression. If the firstName string variable is made only out of letters (either lowercase or uppercase).
5. If it doesn't match the regular expression then return the errors array.

The following picture depicts the work you've done.



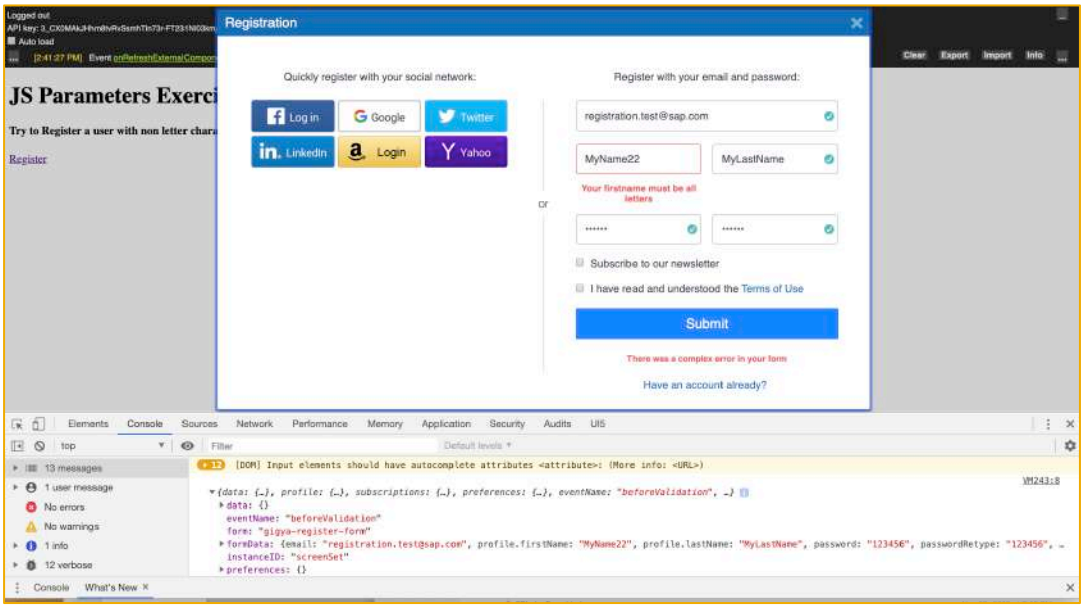
5. Click **Save**.
6. You'll be prompted to create a new version. Enter **a short description** of your changes.
7. Click **Save Version**.



8. Click **X** to exit from the UI Builder.
9. To test the new JavaScript Parameter behavior, create a **new HTML** page with the following code or reuse the registration link from any previous exercise:

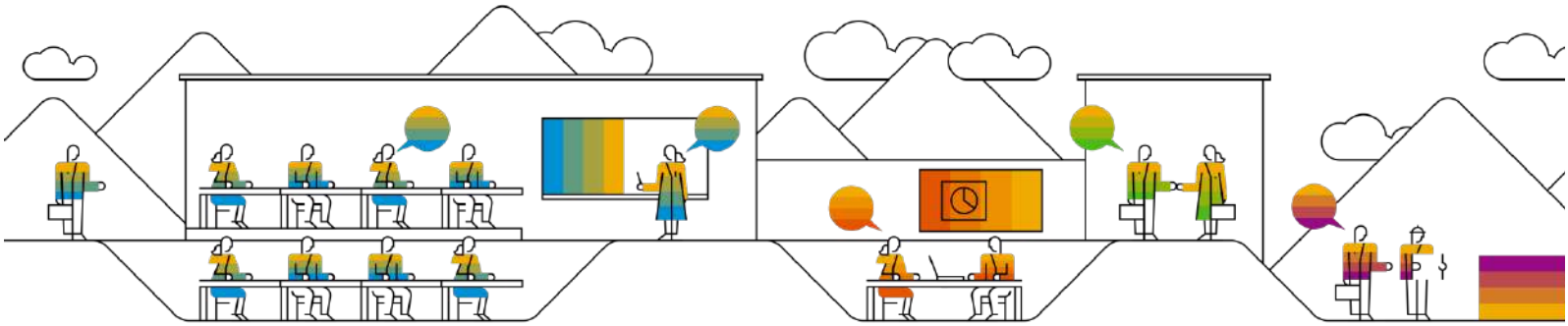
```
<html>
  <head><title>JS Parameters Exercise</title>
    <SCRIPT type="text/javascript" lang="javascript"
            src="http://cdn.gigya.com/js/gigya.js?apikey=YOUR-API-KEY">
    </SCRIPT>
  </head>
  <body><h1>JS Parameters Exercise</h1>
  <h4>Try to Register a user with non letter characters in the first name</h4>
      <a href="#" onclick="gigya.accounts.showScreenSet({screenSet:'Default-
RegistrationLogin', startScreen:'gigya-register-screen'});">Register</a>
  </body>
</html>
```

10. Save the **HTML page** in the root folder of your local web server and navigate to it.
11. Click on the **Register** link
12. Enter a **name with numbers or any other character that is not a letter**.
13. When you click on submit the validation function is triggered and our custom registration logic is evaluated… Here the validation errors will appear:



Recap

In this exercise you learned how to create custom JavaScript Parameters that allow you to plug your custom client-side logic to the Screen-sets lifecycle.



Exercise 9: Dataflows

Create a Customer Data Cloud Dataflow Using the Console

In this exercise, you will access Customer Data Cloud Console and create a Dataflow for doing some Customer data governance operations. The purpose of the exercise is to learn how to create Customer Data Cloud dataflows using the console.

Task 1: Dataflow creation.

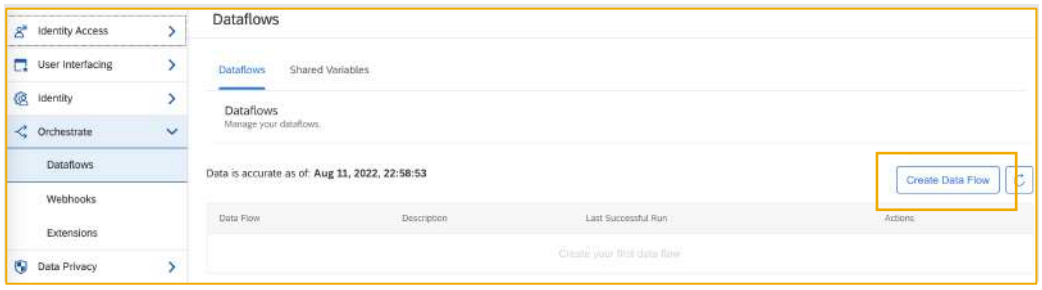
Create a dataflow that is going to process all the user full accounts in the Accounts store.

1. Use **gigya.account** to process the selected records.
2. Use **record.evaluate** to uppercase the first name and last name of each selected account
3. Use **gigya.generic** Update the user record with the *requirePasswordChange* flag set to true in the identity database using a REST API call.

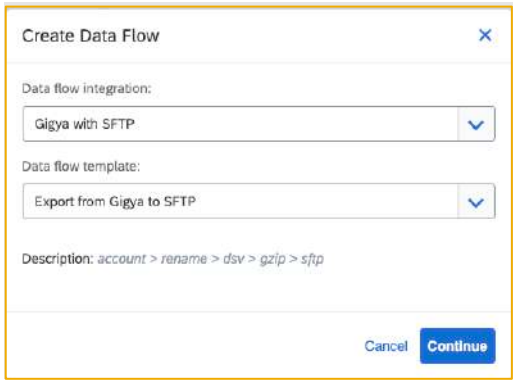
Solution

To create a Customer Data Cloud Dataflow using the console, follow these steps:

1. Login into **CDC's console** and select **Orchestrate** > **Dataflows** from the navigation menu of your site.



2. Click on **Create Data** Flow.
   - Select any two options from the list of templates.



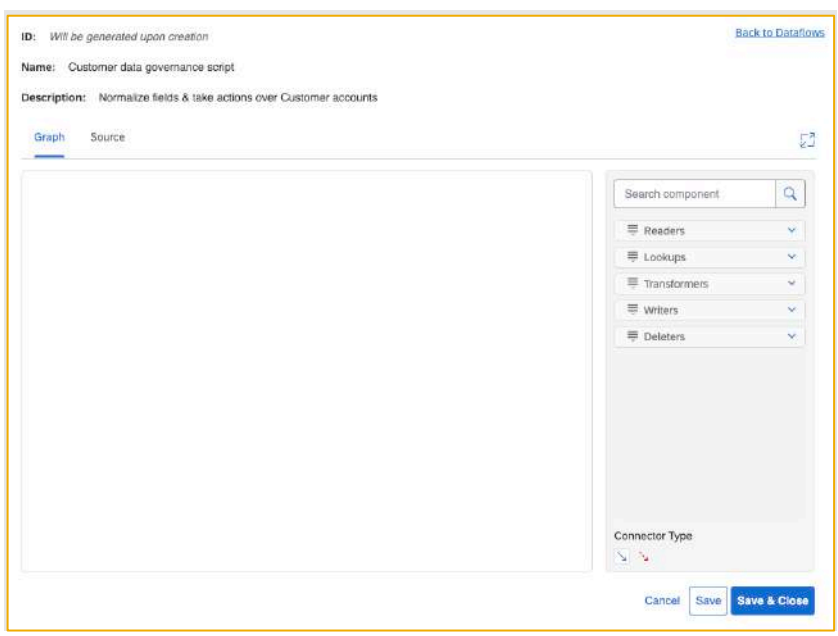
3. Click the **Source** tab for the created dataflow to see the json source codes for it.

4. Remove all the elements of the steps array.
5. Then, adjust its name and description.

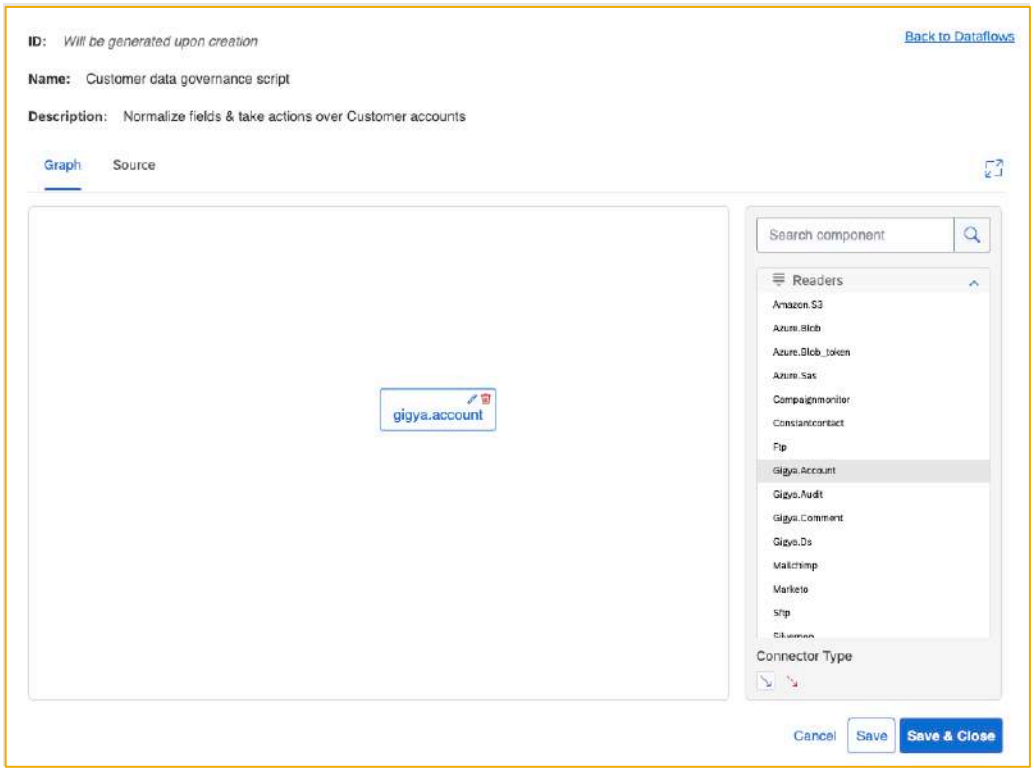
   The dataflow should look like this JSON object:

```
{
 "name": "Customer data governance script",
 "description": "Normalize fields & take actions over Customer accounts",
 "steps": [
 ]
}
```

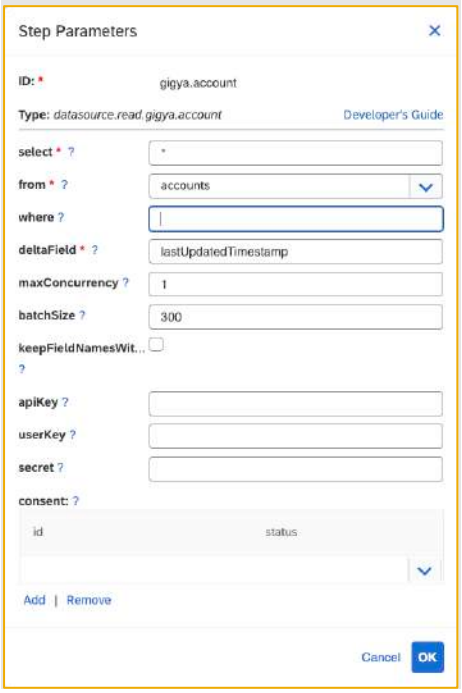
6. Click on the **Graph** tab (you should see an empty script).



7. From the Components menu, select **Readers** > **Gigya.Account** and drag & drop it into your Dataflow design space.

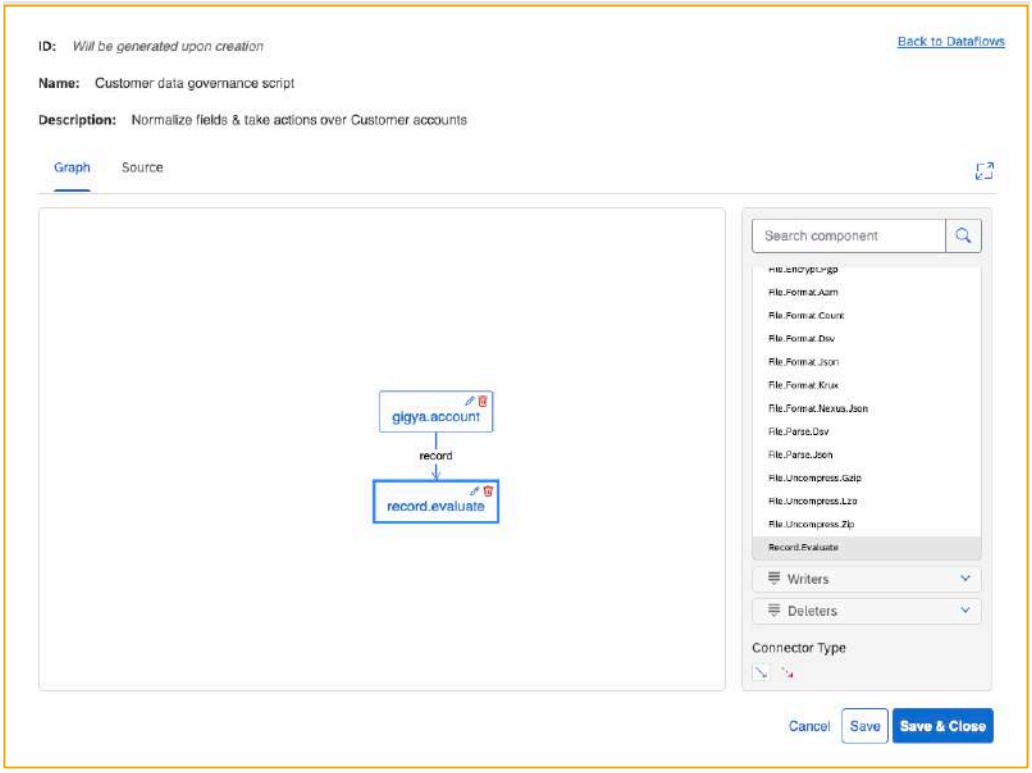


8. Click on the **edit** component properties icon.
9. Add the following properties (add an asterisk **"*"** to Select parameter, to retrieve all the schema fields),
10. Click **OK**.



11. Select **Transformers** > **Record.Evaluate**, drag & drop it into your Dataflow design space.
12. Then, use the **success path** blue arrow to connect with the previous component.

    The final result should look like this.



13. Edit the properties for this **Record.Evaluate** component.
14. Add the following code to the JavaScript function:

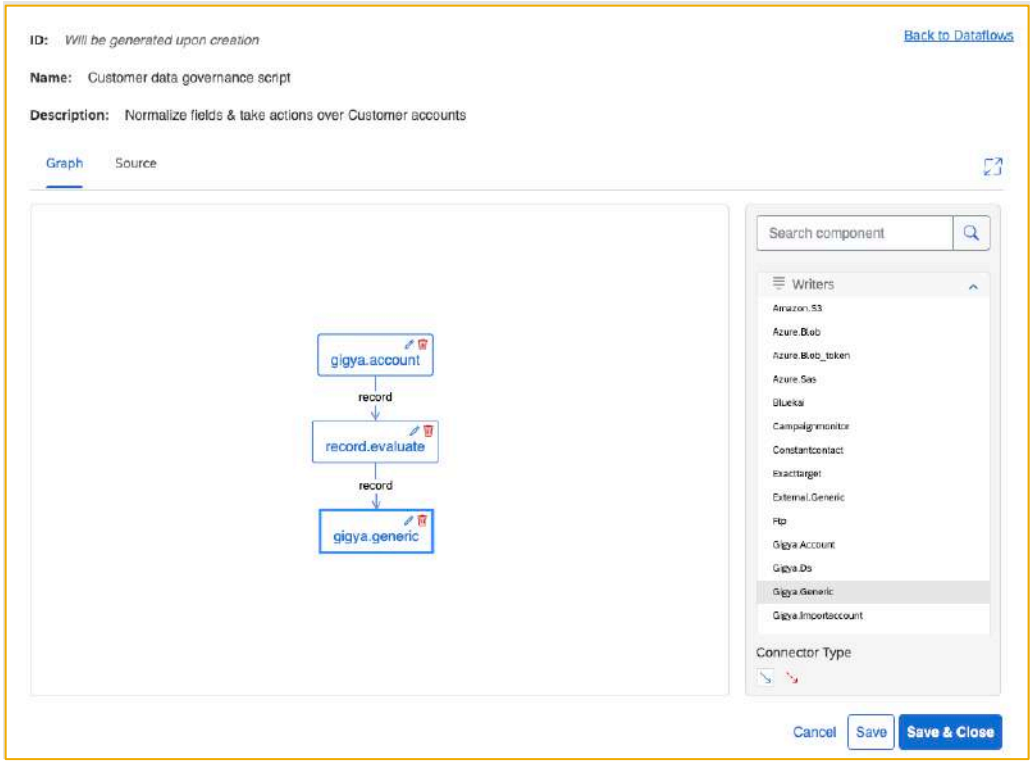
```
function process(record, ctx, logger, next) {
     if (record !== null) {
        if (record.profile !== null && record.profile.firstName !== null
        && record.profile.lastName !== null) {
            logger.info("processing", record.profile.firstName);
            record.profile.firstName=record.profile.firstName.toUpperCase();

      record.profile.lastName=record.profile.lastName.toUpperCase();
            }
     }
     return record;
}
```

15. Click **OK**.

    This function will transform each customer account record by setting the first and last name of the customer in capital letters. Also, the function logs each record processed.

16. Select ***Writers*** > ***Gigya.Generic*** component and drag & drop.
17. Then, use the **success path** blue arrow to connect with the previous component.

    The final result should be like this:



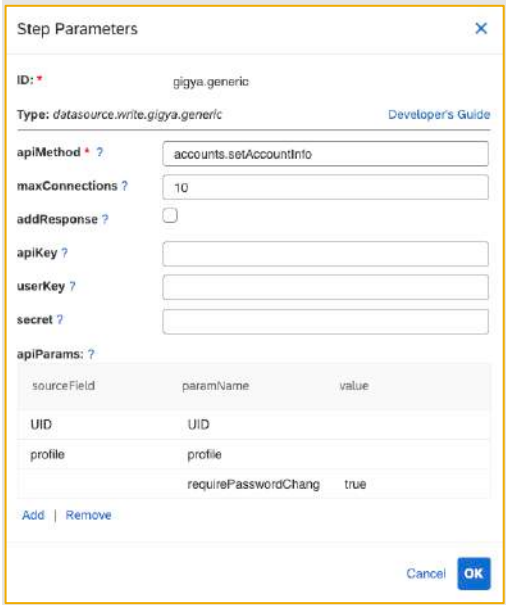
18. Click on the **edit** component properties icon and add the following properties:

    apiMethod: *accounts.setAccountInfo*
    apiParams

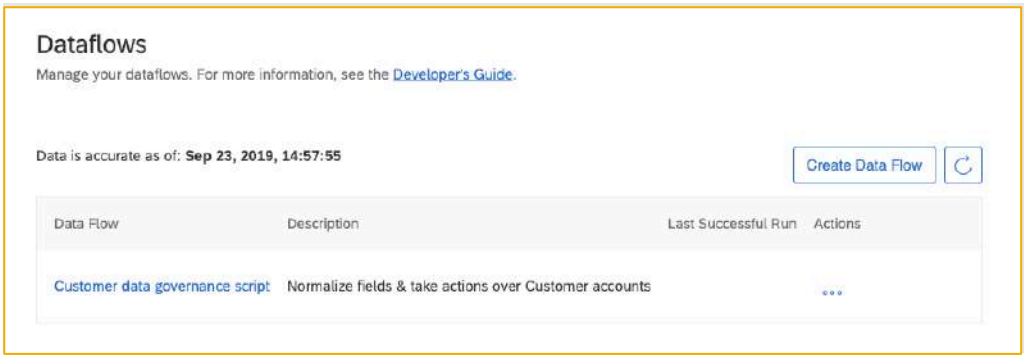
sourceField	paramName	value
UID	*UID*	
profile	*profile*	
	requirePasswordChange	*true*

Refer to the [API documentation]() for the type and description of each parameter.

It should look like this:



19. Then, click **OK**.
20. Click on **Save & Close**.
    The new dataflow should look like this:



Recap

In this exercise, you used one of the CDC's extensibility mechanisms JavaScript Parameters. You customized the customer registration process with custom codes running in client-side browsers.

Task 2: Dataflow Testing.

Test the dataflow and notice the execution results.

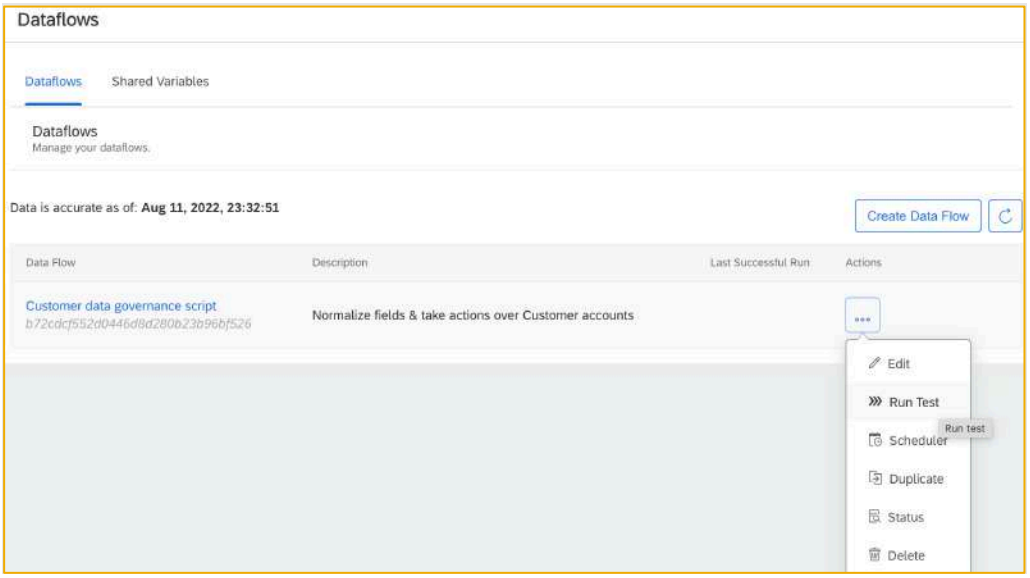
Solution

To test the dataflow, follow these steps:

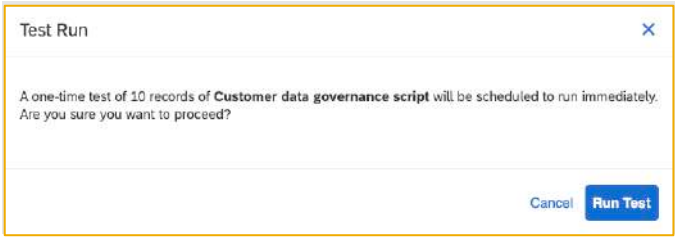
1. Make sure you have at least one full customer account within your site. Go to **Identity Access** and check the **customer records**.

   You should see something like this:

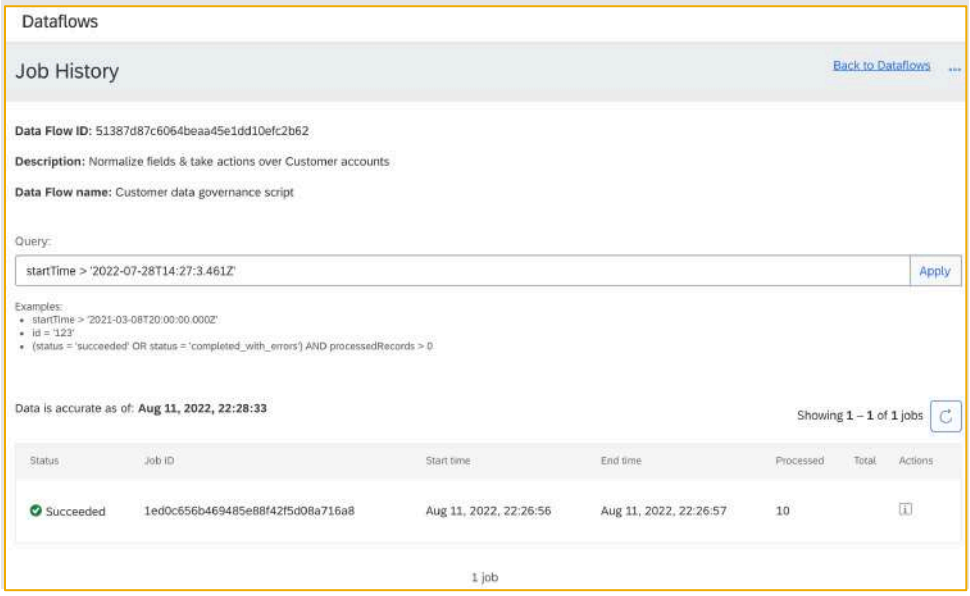
2. Go back to the navigation menu and select ***Orchestrate*** > ***Dataflows***.
3. Then, click on the **three dots** that appear on the right-hand side of your dataflow and select **Run Test**.



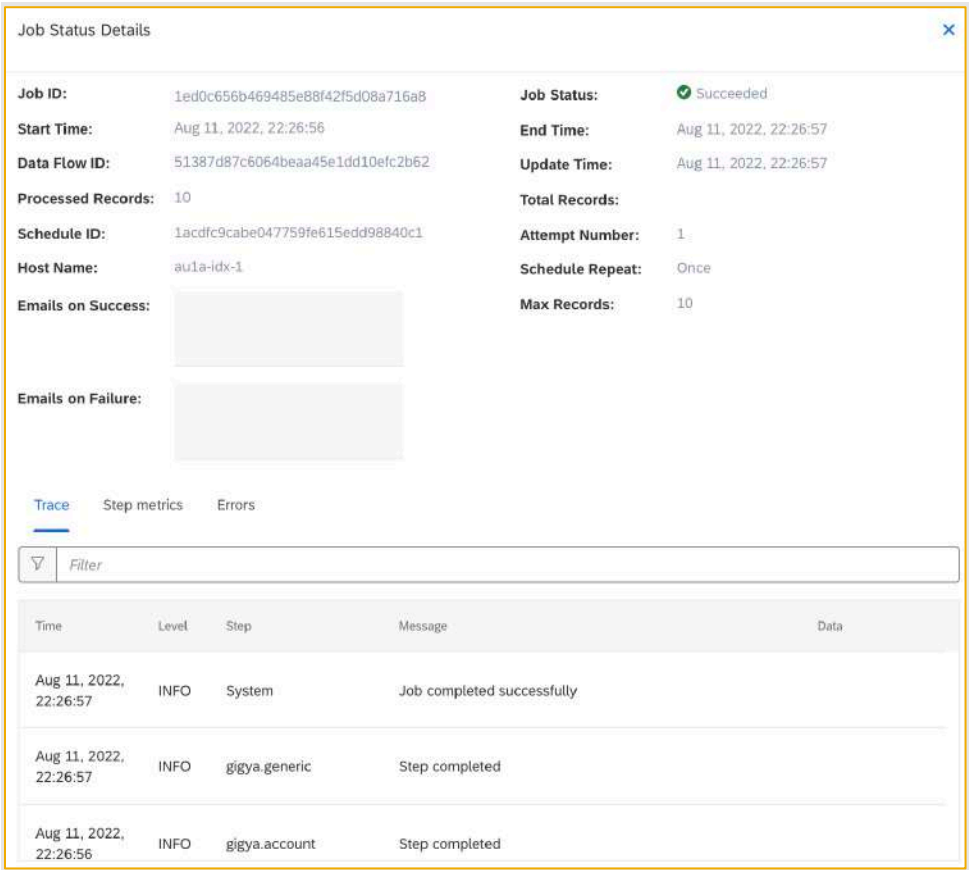
4. Click **Run Test** to accept the dialog the appears.



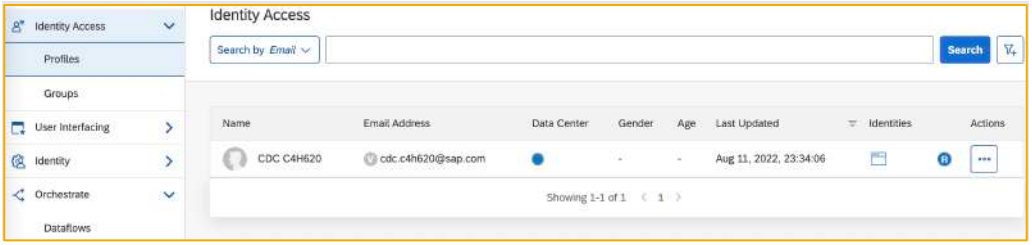
5. Click again on the **three dots** and select **Status**. After few seconds you should see that the dataflow has been run by the system.



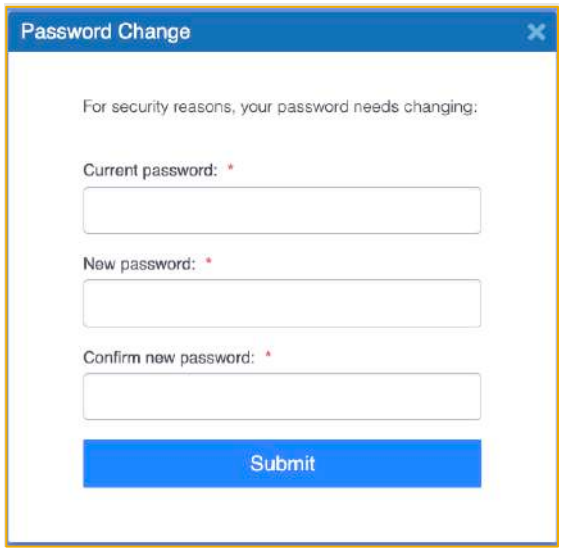
6. Select the **Information** action (the “i” icon).
7. Select the **Job Status Details**.
8. Select the **Trace** information tab.
9. Select the **Step metrics** and **Errors** tabs.
10. Click on the **Close** button.



11. Verify the script result.
    - Go back to the **Identity Access** console tool.
    - Check the list of customer accounts. The Name and Surname must be in uppercase letters.



OPTIONAL. If you try to login to the customer using the screen sets you should see the password change screen right after login the user.



Recap

In this exercise, you tested the dataflow to check for execution results.

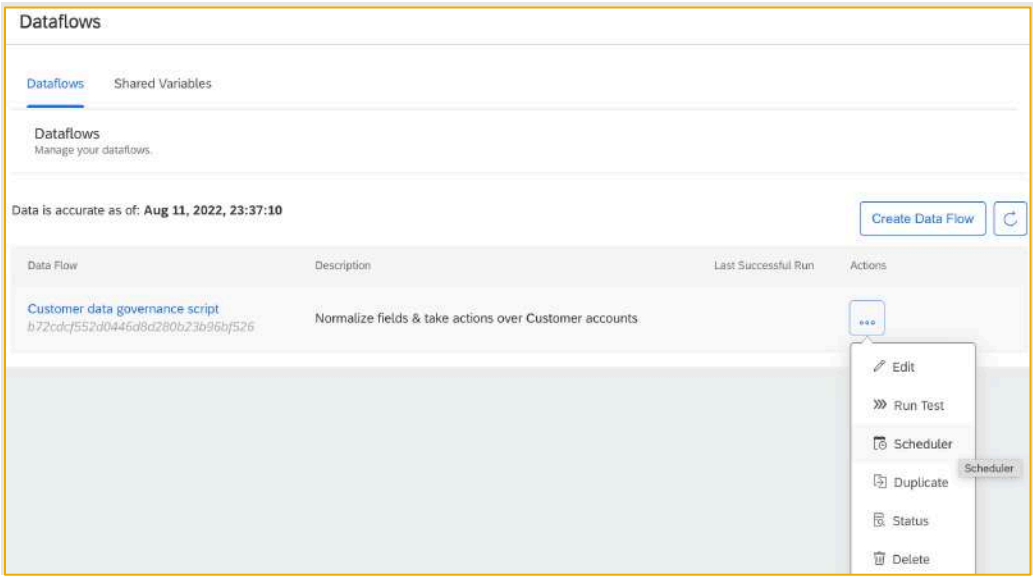
Task 3: Dataflow Scheduling.

Create and schedule for running it at midnight every day.

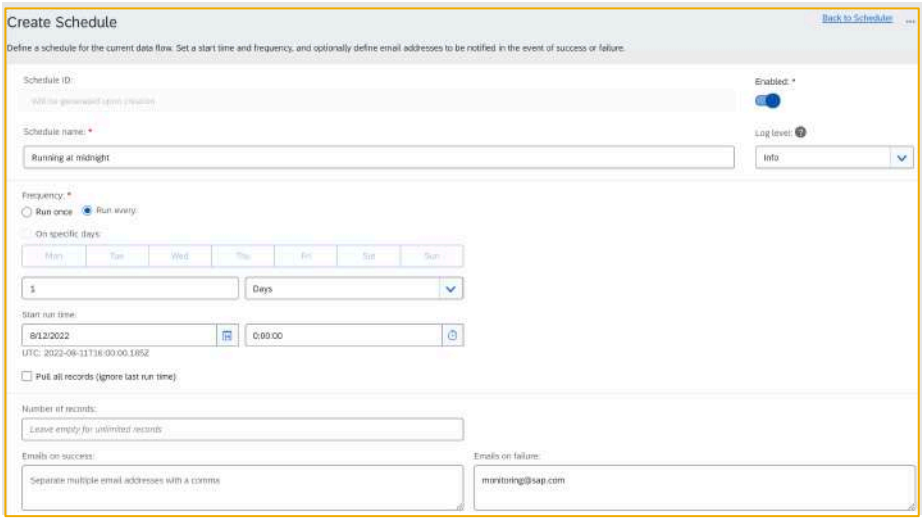
Solution

To create and schedule for this dataflow, follow these steps:

1. Go back to the navigation menu and select **Orchestrate > Dataflows**.
2. Click on the **three dots** action for your dataflow and select **Scheduler**.



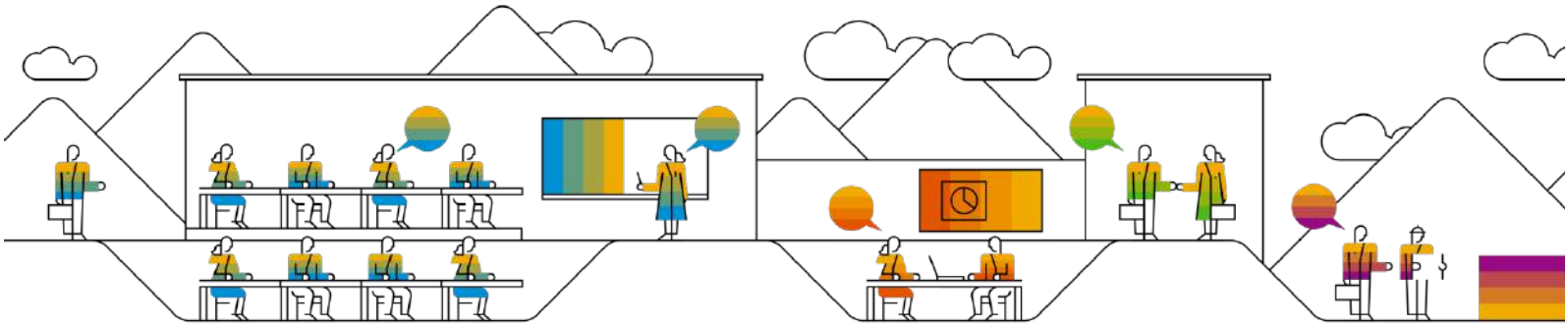
3. Click on **Create Schedule**.
4. Enter the following information for your schedule:
   - Name
   - Start run time (midnight)
   - Enabled
   - Frequency
   - Emails on failure etc.
5. Click on **Create**.



6. You have now created the schedule for this Dataflow. Come back here after midnight to check the status of the Schedule again.

Recap

In this exercise you learned how to create, test, and schedule Dataflows. You have also used in the Dataflow script some of the most helpful components.



Exercise 10: JSON Web Tokens (JWT)

Request Customer Data fields using REST API

In this exercise you will use one of the SAP Customer Data Cloud's data sharing mechanisms, JSON Web Tokens (JWT). You will learn how to request the customer data fields using the REST API and get the public key to validate them.

Task 1: JWT processing using the REST API

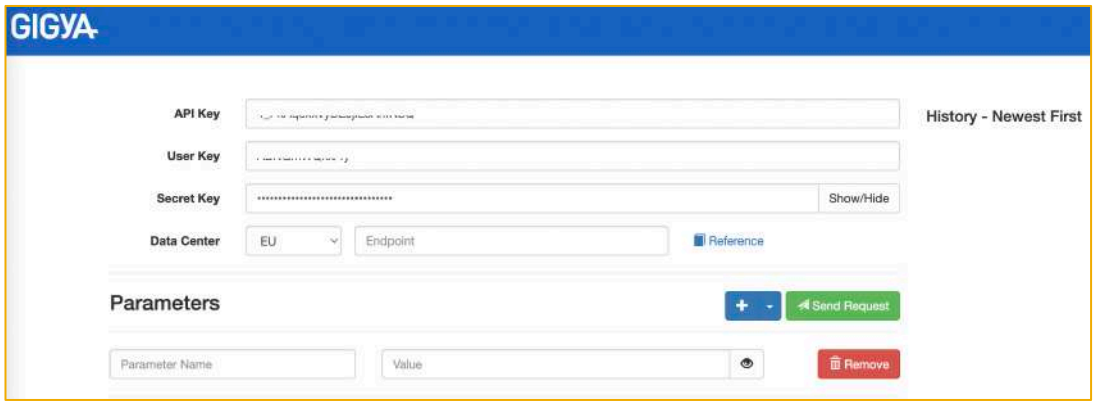
Using the REST API, rquest the ID token for one of the registered customers in CDC database. Ask for first name, last name, and email fields. Then, get the public key from the server and validate that the information hasn't been tampered.

> **Tip**: Use https://jwt.io debugger to validate the id token against the public key.

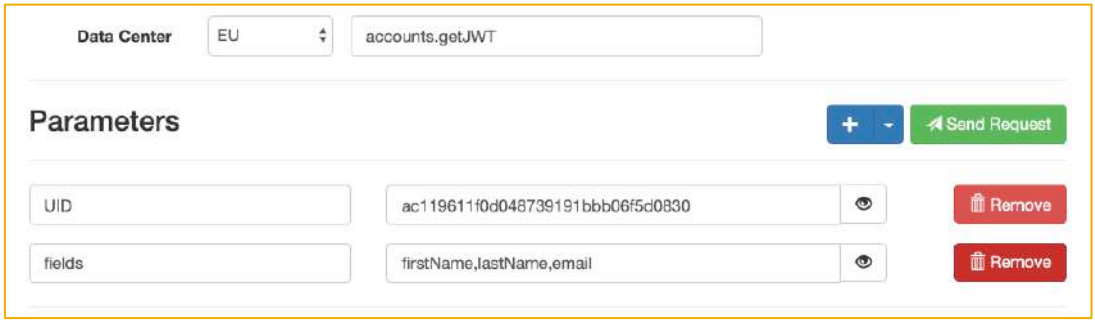
Solution

To request Customer Data fields using REST API, follow these steps:

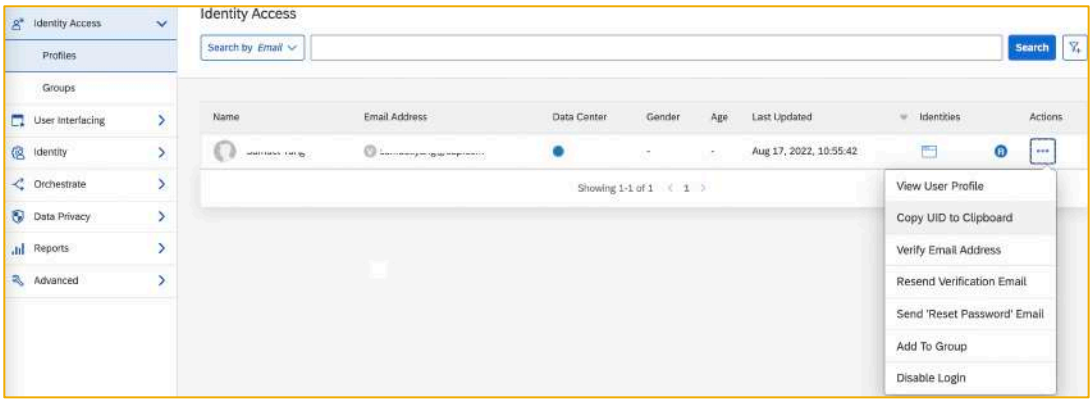
1. Open **Gigya API TOOL** https://tools.gigya-cs.com/api/ with your browser.
2. Enter the following:
   - Your API Key
   - Your User Key
   - Your Secret Key
   - Your site data center.



3. Enter the **accounts.getJWT** as the endpoint with the **UID** and **fields** parameters .



4. To get the UID for any of the customers in your site accounts database:
   - Go to CDC Console.
   - Select Identity Access > Profiles.
   - Click on the **three dots** under the Actions column.
   - Select **Copy UID to Clipboard**.



5. Click on **Send Request,** on the center of the right-hand side of the screen and check the CDC server response results. Notice the **id_token** attribute, as long as you get no errors:

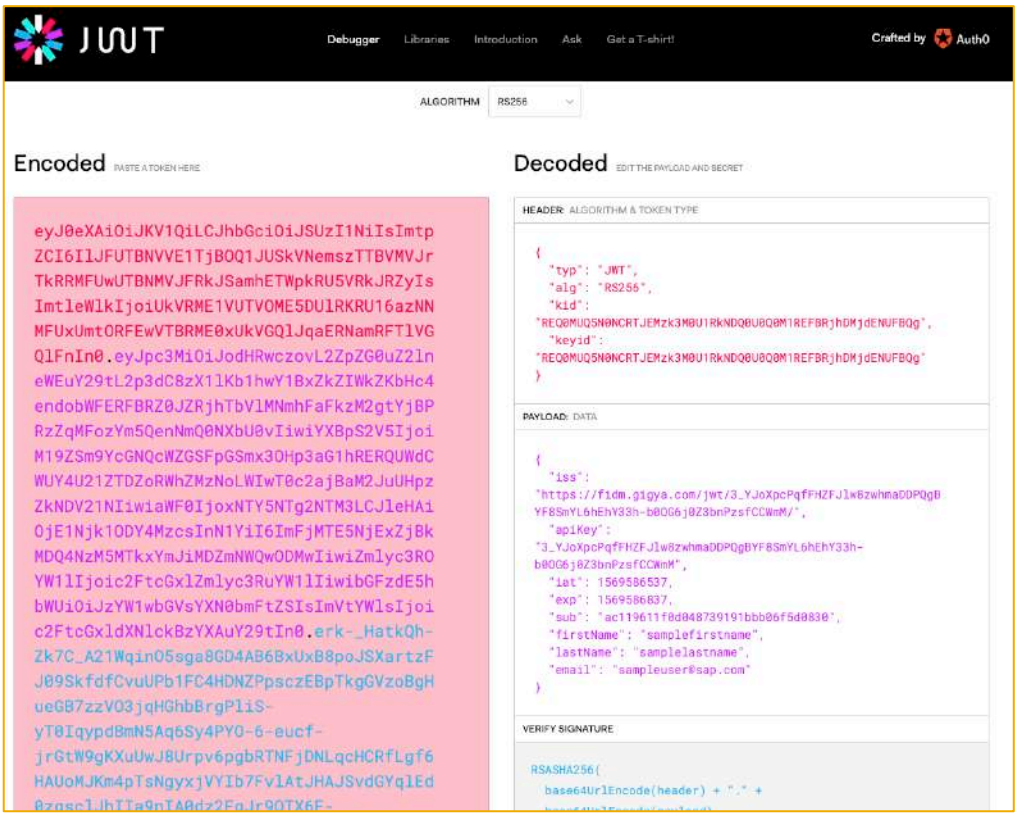
6. Select the **id_token** value and **copy** it, (make sure you select the complete string).

   Your results should look like this:

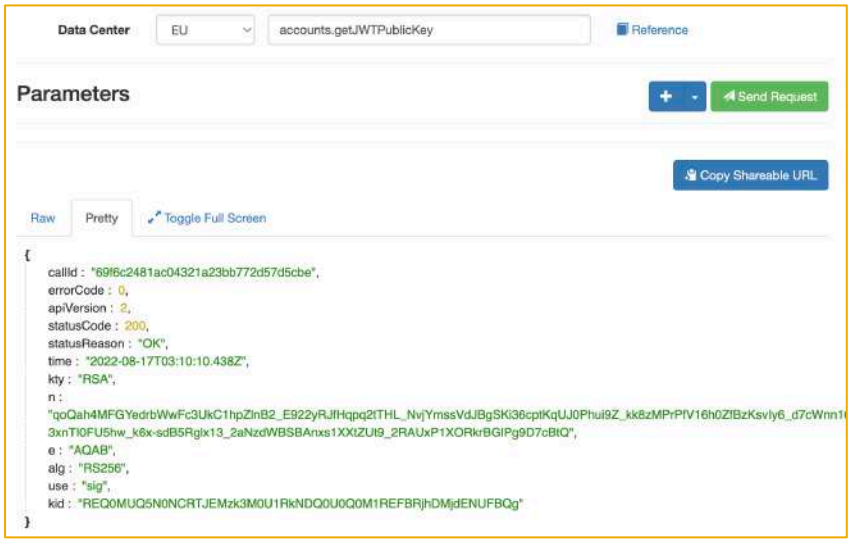

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IlJFUTBNVVE1TjBOQ1JUSkVNemszTT
BVMVJrTkRRMFUwUTBNMVJFRkJSamhETWpkRU5VRkJRZyJ9.eyJpc3MiOiJodHRwczovL2ZpZG0uZ2lne
WEuY29tL2p3dC80X1BrQXFjeHh2eURaSmppTDhBaGZOYlEvIiwiYXBpS2V5IjoiNF9Qa0FxY3h4dnlEW
kpqaUw4QWhmTmJRIiwiaWF0IjoxNjYwNzA1MjQwLCJleHAiOjE2NjA3MDU1NDAsInN1YiI6IjhiOTk0M
zU0YzRhNzQzOGE4YjhhZGRhZTgyOTY0NzY1IiwiZmlyc3ROYW1lIjoiU2FtdWVsIiwibGFzdE5hbWUiO
iJZYW5nIiwiZW1haWwiOiJzYW11ZWwueWFuZ0BzYXAuY29tIn0.OILQP431upsJTYfb_zsYzT5k3-
KFOmnzVLRosd5fmRrD5gyCF6tVSJ7_V1aqBeHAKA1HYtPZzoq2-
TcwTzDgx4BEEXztWtV0ZzHmOQgipocw0GuoiDv3eJIsobRS4JnBholoTxQt4AKFVSZ_n2qFqFbukIwMt
N8pb7XtduBJpdb2XLmzNWTz3M_HvE3LogmMbnSNcddGurIXFbl5pGANGmjKPdpf9-mKiyZGX3G-
x6FWMpqH1CmbldqFJNntA1IGlQqBoCUxkC3nZ8VHS_a4pys2UDX5nu2AZnHOnlxQTy3ICerL4XlX-
wFbfIMWEU3cLsSOhv2BddMVSGBkcQrXAQ
```


7. Navigate with your browser to [https://jwt.io.](https://jwt.io)

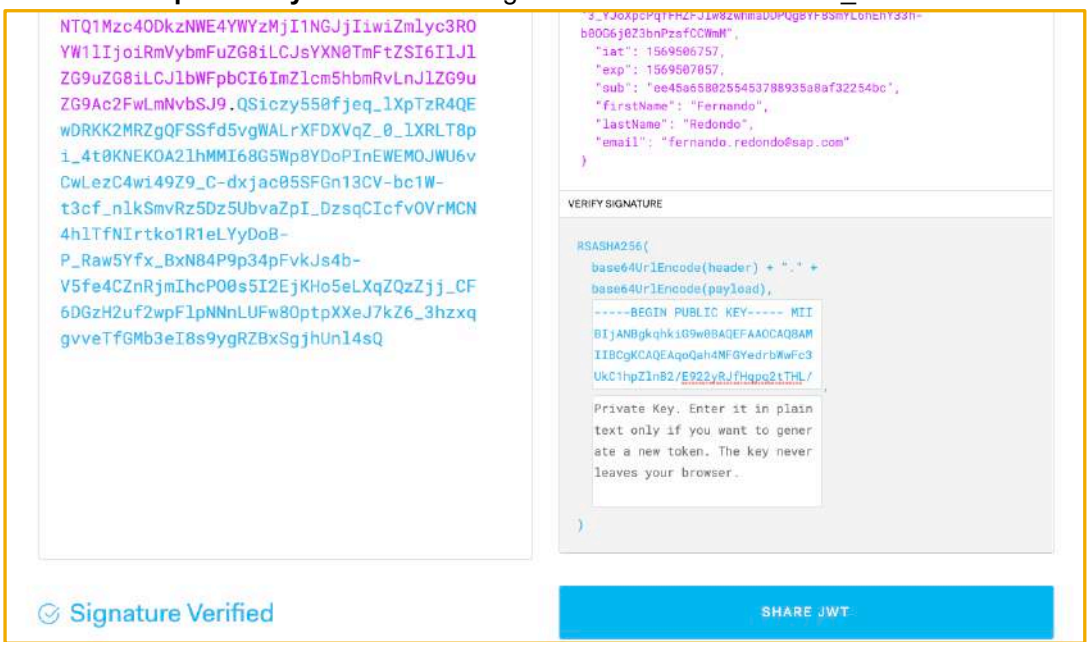
8. Paste in the **Debugger** the *id_token* you previously selected.



9. Notice the Encoded area at the left-hand side with the id_token raw information just **colorized** for the 3 different parts of it: Header, Payload and Signature. The same information is BASE64 decoded at the right-hand side showing the proper JSON objects that compose the id_token. Take a moment to check the payload fields, including those you have specifically requested (firstName, lastName and email).
10. Go back to the **Gigya API TOOL**.
11. Create a new request pointing to the **accounts.getJWTPublicKey** endpoint. As this request has no parameter.
12. Click on **Remove** to remove the existing two parameters from the previous request.



13. Copy the **resulting payload** (the whole JSON string including the curly braces) from the previous step.
14. Paste it into the **first (Public Key) VERIFY SIGNATURE** textbox on the jwt.io debugger page.
15. Use that **RSA public key** to check if the signature of the JWT for the id_ token was valid.



Recap

In this exercise you learned how to request JWT tokens (id_tokens) from the CDC server and how to validate its signature using the JWT public keys.