C4H620

SAP Customer Data Cloud Developer

PARTICIPANT HANDBOOK
INSTRUCTOR-LED TRAINING

Course Version: 2008
Course Duration: 3 Day(s)
Material Number: 50155275

SAP Copyrights, Trademarks and Disclaimers

Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation	🖥
Demonstration	➤
Procedure	1 2 3
Warning or Caution	⚠
Hint	💡
Related or Additional Information	»
Facilitated Discussion	💬
User interface control	*Example text*
Window title	*Example text*

Contents

Course Overview

TARGET AUDIENCE

This course is intended for the following audiences:

- Application Consultant
- Development Consultant
- Developer

UNIT 1 Solution Overview

UNIT OBJECTIVES

- Explain the basics of the SAP Customer Data Cloud (CDC) platform and API

Unit 1
Lesson 1

Introducing SAP Customer Data Cloud

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the basics of the SAP Customer Data Cloud (CDC) platform and API

An Introduction to the Platform and API

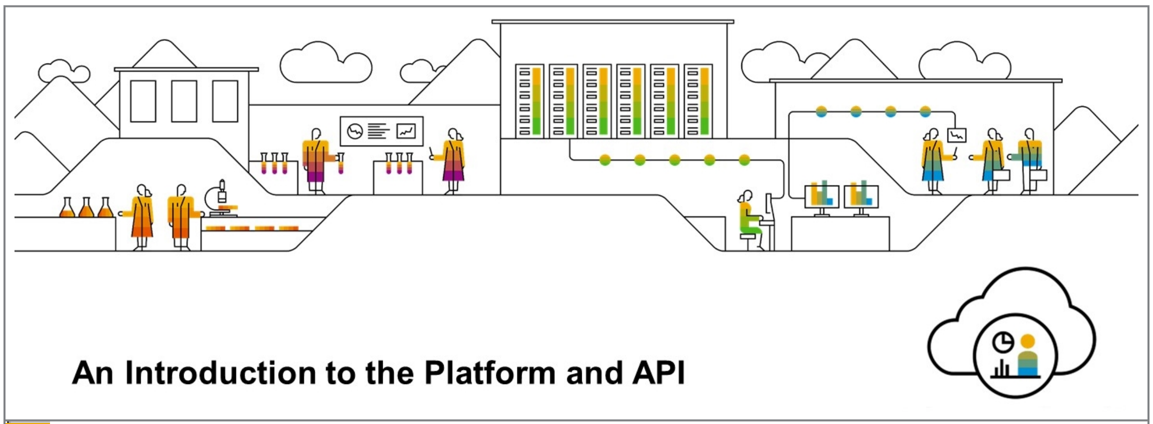


Figure 1: An Introduction to the Platform and API

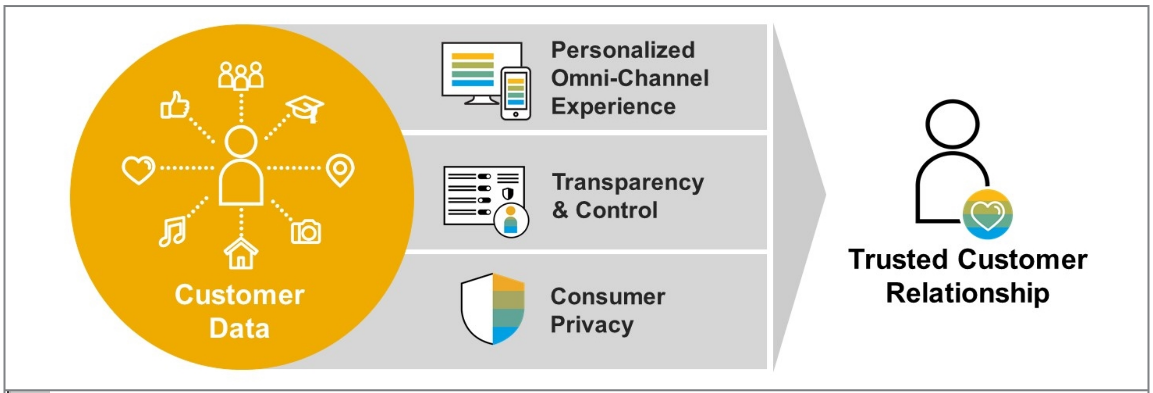

Customer Data
Personalized Omni-Channel Experience
Transparency & Control
Consumer Privacy
Trusted Customer Relationship

Figure 2: Megatrends driving the need to build trusted customer relationships

Customer Experience pillars:

1. Know your customer, have a 360 view
2. Build trust with customers / deliver value in exchange for data / companies must become more sophisticated in the way they accumulate and manage personal data
3. Deliver relevant and convenient services and products

4. Regulations are shifting the power to control personal data collection and usage back to the consumer, companies must be compliant / companies must seize this moment to earn trust for their benefit

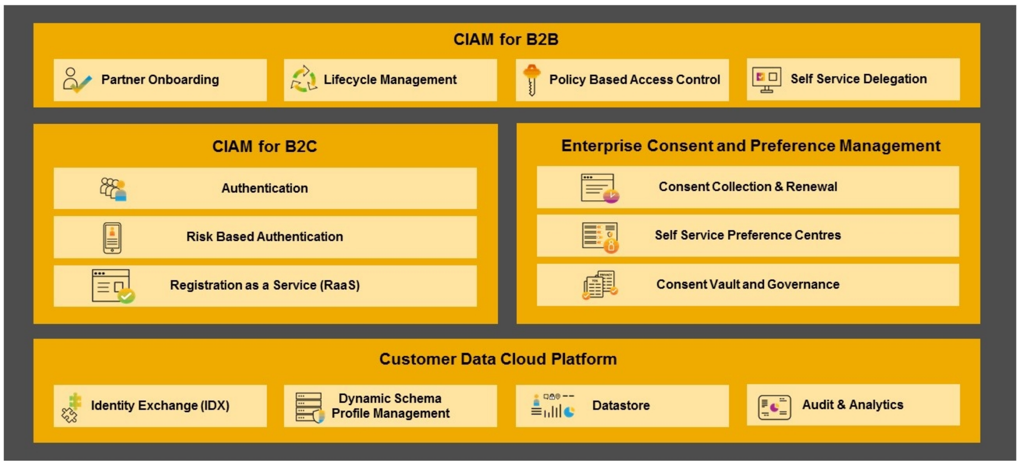

CIAM for B2B
Partner Onboarding
Lifecycle Management
Policy Based Access Control
Self Service Delegation
CIAM for B2C
Authentication
Risk Based Authentication
Registration as a Service (RaaS)
Enterprise Consent and Preference Management
Consent Collection & Renewal
Self Service Preference Centres
Consent Vault and Governance
Customer Data Cloud Platform
Identity Exchange (IDX)
Dynamic Schema Profile Management
Datastore
Audit & Analytics

Figure 3: SAP Customer Data Cloud Solutions

CDC will be used to refer to SAP Customer Data Cloud throughout the training materials.

CIAM stands for Customer Identity and Access Management.

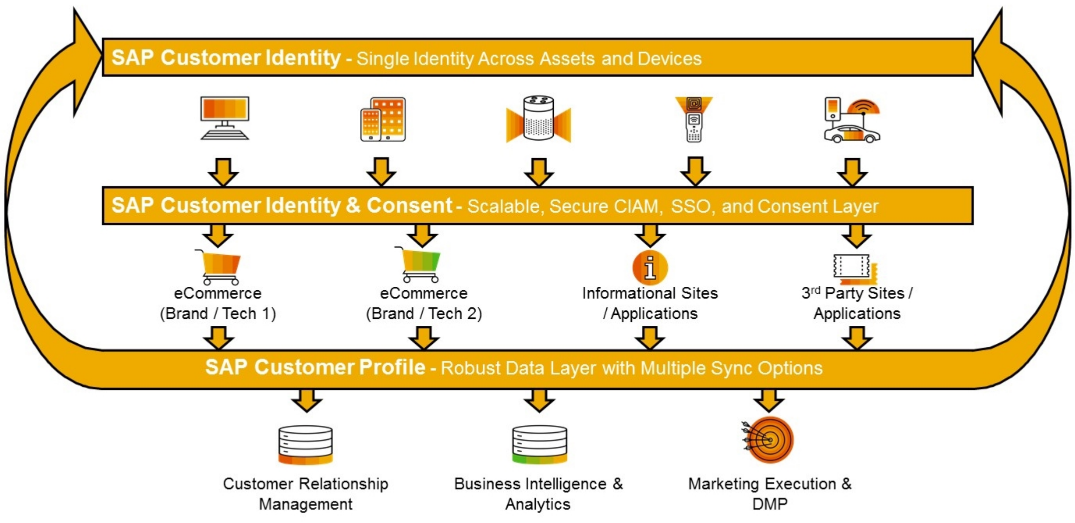

SAP Customer Identity - Single Identity Across Assets and Devices
SAP Customer Identity & Consent - Scalable, Secure CIAM, SSO, and Consent Layer
eCommerce (Brand / Tech 1)
eCommerce (Brand / Tech 2)
Informational Sites / Applications
3rd Party Sites / Applications
SAP Customer Profile - Robust Data Layer with Multiple Sync Options
Customer Relationship Management
Business Intelligence & Analytics
Marketing Execution & DMP

Figure 4: Customer Data Cloud: A robust, scalable Identity and Consent platform

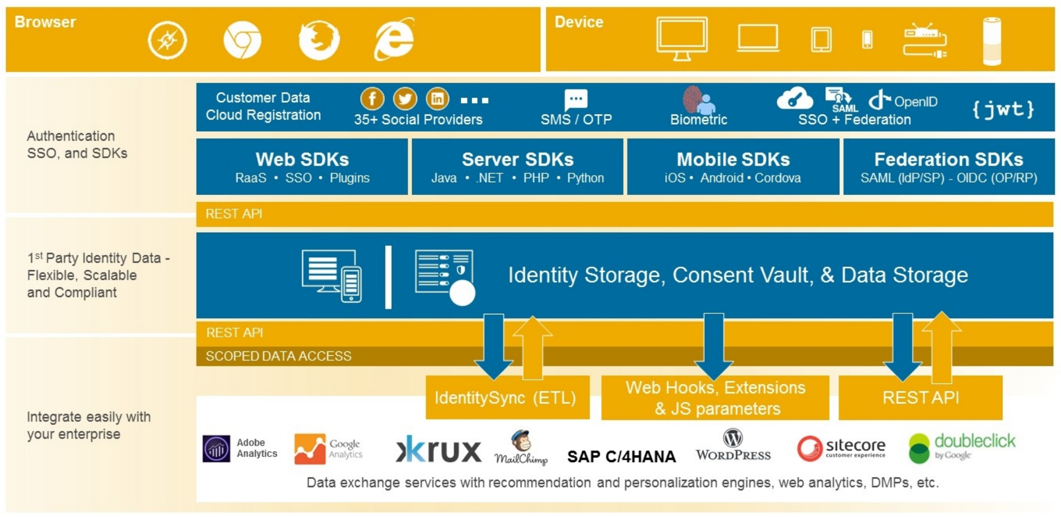
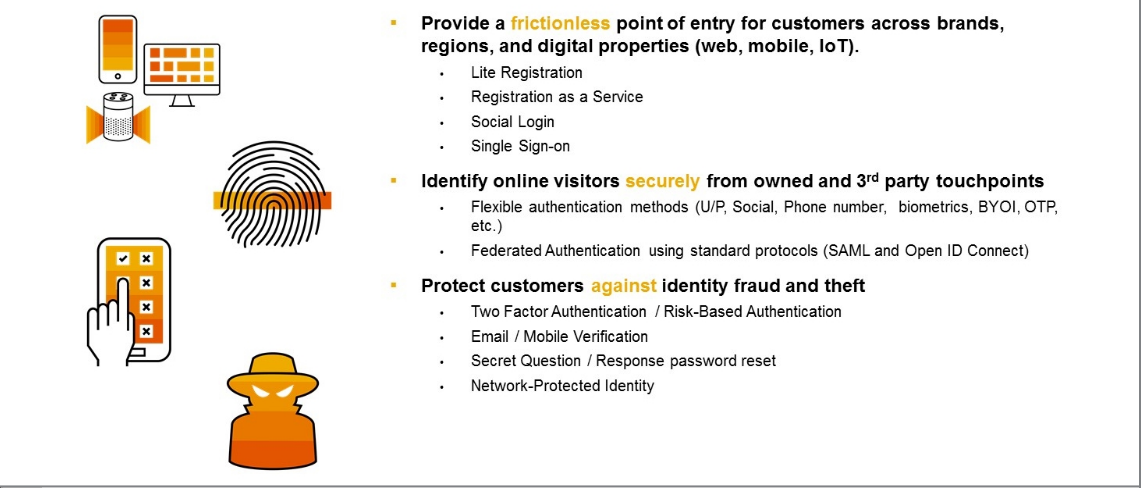

Browser
Device
Authentication SSO, and SDKs
Customer Data Cloud Registration
35+ Social Providers
SMS / OTP
Biometric
SAML
OpenID
SSO + Federation
{jwt}
Web SDKs
RaaS • SSO • Plugins
Server SDKs
Java • .NET • PHP • Python
Mobile SDKs
iOS • Android • Cordova
Federation SDKs
SAML (IdP/SP) - OIDC (OP/RP)
REST API
1st Party Identity Data - Flexible, Scalable and Compliant
Identity Storage, Consent Vault, & Data Storage
REST API
SCOPED DATA ACCESS
IdentitySync (ETL)
Web Hooks, Extensions & JS parameters
REST API
Integrate easily with your enterprise
Adobe Analytics
Google Analytics
krux
MailChimp
SAP C/4HANA
WordPress
sitecore
doubleclick by Google
Data exchange services with recommendation and personalization engines, web analytics, DMPs, etc.

Figure 5: The Customer Data Cloud Platform - Overview



- **Provide a frictionless point of entry for customers across brands, regions, and digital properties (web, mobile, IoT).**
  - Lite Registration
  - Registration as a Service
  - Social Login
  - Single Sign-on
- **Identify online visitors securely from owned and 3rd party touchpoints**
  - Flexible authentication methods (U/P, Social, Phone number, biometrics, BYOI, OTP, etc.)
  - Federated Authentication using standard protocols (SAML and Open ID Connect)
- **Protect customers against identity fraud and theft**
  - Two Factor Authentication / Risk-Based Authentication
  - Email / Mobile Verification
  - Secret Question / Response password reset
  - Network-Protected Identity

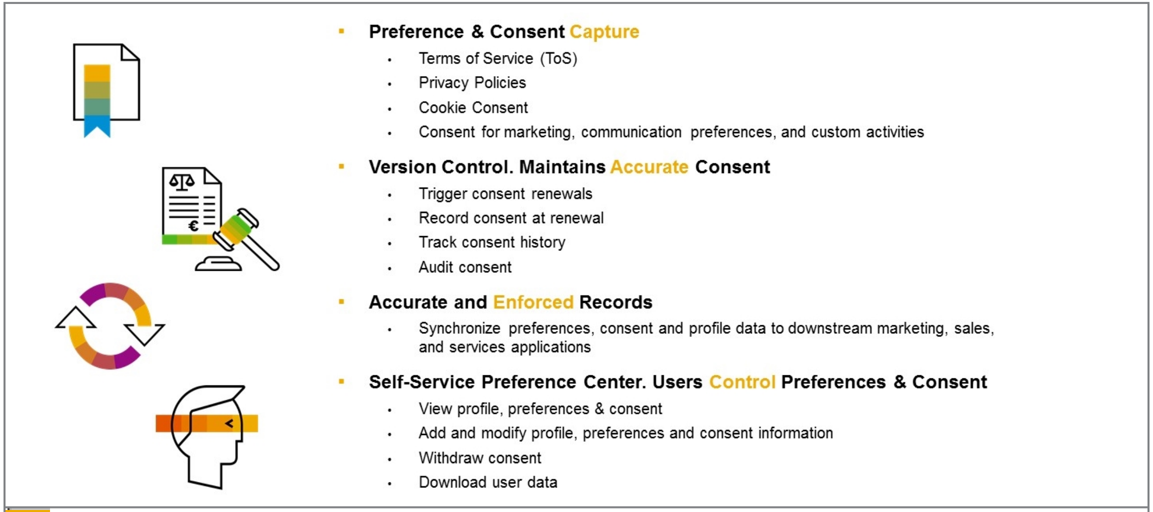
Figure 6: SAP Customer Identity - Identify customers across channels & devices

First module of the solution - SAP Customer Identity

1. Provides a frictionless point of entry for customers across brands, regions, and properties, with customizable and secure registration and social login screens and flows. U/P: Username and Password, BYOI: bring your own identity, OTP: One Time Password.

2. Securely identifies online visitors from any touchpoint using federation, single sign-on, and flexible user authentication options. https://developers.gigya.com/display/GD/Authentication+Options

3. Provides many security features against identity fraud and theft, such as increased security levels based on context (RBA-Risk Based Authentication) or account takeover protection.

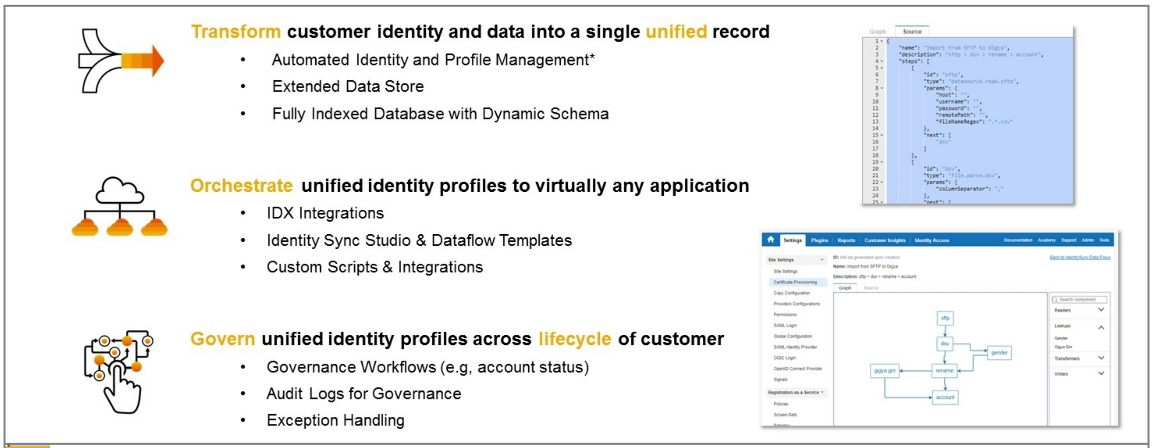


- **Preference & Consent Capture**
  - Terms of Service (ToS)
  - Privacy Policies
  - Cookie Consent
  - Consent for marketing, communication preferences, and custom activities
- **Version Control. Maintains Accurate Consent**
  - Trigger consent renewals
  - Record consent at renewal
  - Track consent history
  - Audit consent
- **Accurate and Enforced Records**
  - Synchronize preferences, consent and profile data to downstream marketing, sales, and services applications
- **Self-Service Preference Center. Users Control Preferences & Consent**
  - View profile, preferences & consent
  - Add and modify profile, preferences and consent information
  - Withdraw consent
  - Download user data

Figure 7: SAP Customer Consent - Managing preferences & consent for customers

Second module of the solution - Customer Consent

1. SAP Customer Consent helps businesses manage consent throughout the lifecycle of the customer.
2. Consent flow
   a. i. Present and capture consent
      ii. Maintain accurate consent, including auto-prompts for consent when policies are changed
      iii. Enforce consent when syncing data outside of Customer Data Cloud
      iv. Provide control to users for their data - view, update, delete, ...



Transform customer identity and data into a single unified record
- Automated Identity and Profile Management*
- Extended Data Store
- Fully Indexed Database with Dynamic Schema

Orchestrate unified identity profiles to virtually any application
- IDX Integrations
- Identity Sync Studio & Dataflow Templates
- Custom Scripts & Integrations

Govern unified identity profiles across lifecycle of customer
- Governance Workflows (e.g, account status)
- Audit Logs for Governance
- Exception Handling

```
{
    "name": "Import from SFTP to Gigya",
    "description": "sftp > dsv > rename > account",
    "steps": [
        {
            "id": "sftp",
            "type": "datasource.read.sftp",
            "params": {
                "host": "",
                "username": "",
                "password": "",
                "remotePath": "",
                "fileNameRegex": ".*.csv"
            },
            "next": [
                "dsv"
            ]
        },
        {
            "id": "dsv",
            "type": "file.parse.dsv",
            "params": {
                "columnSeparator": ","
            },
            "next": [
```

ID: Will be generated upon creation
Name: Import from SFTP to Gigya
Description: sftp > dsv > rename > account

Figure 8: SAP Customer Profile - Power trusted digital experiences with first-party data

Third and last module of the solution - SAP Customer Profile

1. Leverage CDC's powerful database (extensible data model, fully indexed, ...) to build a unified reference for users' identity data
2. Benefit from CDC's integration capabilities to orchestrate user data exchanges with the rest of the Customer Experience Suite.

3. Identity Sync (dedicated ETL infrastructure + dataflow studio script editor)
4. Comprehensive API
5. WebHooks
6. Identity Governance - with features such as account freezing, auditing capabilities, etc.

Position in the Stack

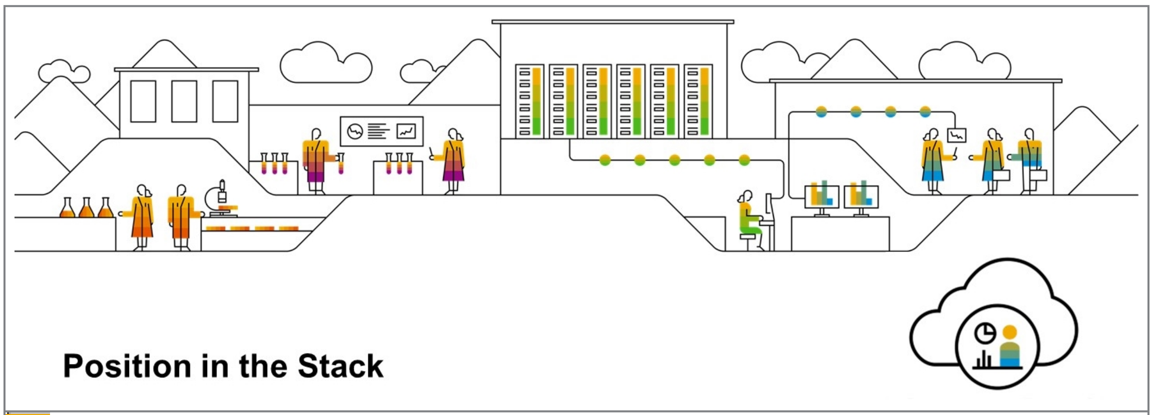


Figure 9: Position in the Stack

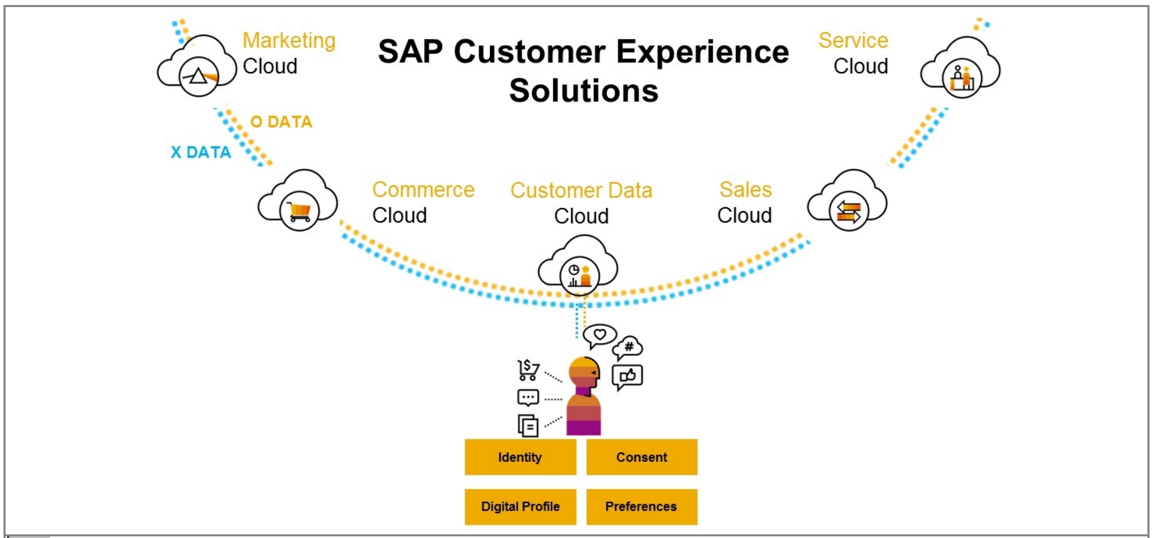


Figure 10: SAP C/4HANA - The Customer Experience Suite

SAP Customer Data Cloud allows storing X (experience) and O (operational) data directly related to your B2C and B2B customers. Identities can be sourced through standard federation mechanisms, such as SAML and OpenID Connect systems, directly through RaaS (site identity) or connected through social networks (we support 35+ social networks in the East and West). Consent and preferences allow your systems to store communication preferences, terms of service, data privacy, and other types of consent, and most importantly, be compliant with data regulation laws. The digital profile can source customer behavioral data from external systems, and also store first party profile information. You can leverage standard integration when connecting SAP Customer Data to other SAP Clouds such as Marketing, Commerce, Sales and Service using GConnectors or IdentitySync to downstream/upstream customer data from/to SAP Customer Data Cloud.

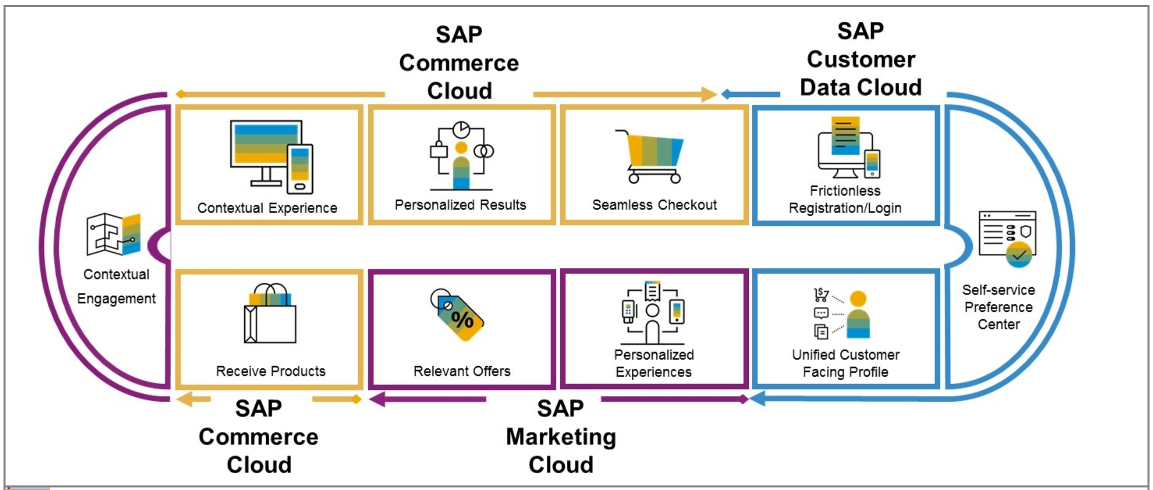


Figure 11: Trusted personalized relationships drive higher customer lifetime value

You can read the circular diagram as follows:

1. A customer decides to shop for products for the first time on an SAP Commerce System connected to SAP Customer Data Cloud. So she/he decides to register a new customer account (Frictionless Registration/Login).
2. The same customer decides to provide extra information using Self-service Preference Center.
3. The customer profile is enriched through other existing profile sources on the connected system, giving the system a Unified Customer Facing Profile.
4. The information is downstreamed from SAP Customer Data Cloud to SAP Marketing Cloud, where the customer can be Segmented (profiled), allowing for Personalized Experiences. The decision of profiling a particular customer is given through affirmative action using SAP Customer Data Cloud Consent and Enterprise Preference Management.
5. SAP Marketing can suggest Relevant Offers (product suggestions) to the customer, always observing his/her consent preferences.
6. SAP Commerce Cloud starts to display suggested products through its integration to SAP Marketing Cloud.
7. SAP Marketing Cloud receives from SAP Commerce Cloud any customer interactions through Contextual Engagement, in order to evaluate how effectively the personalization and customization is reaching the customer.
8. SAP Commerce Cloud can display different banners or content according to the customized personalizations you decide to define using your Target audience (a rule-based mix of your Segments).
9. Also, you can provide different search results to your Target audience using SAP Commerce Adaptive Search.
10. By using SAP Customer Data embedded to SAP Commerce Cloud you can leverage many ways to identify your customer: email, phone, FaceId, TouchId, push notification, social login, SAML, OpenID, etc. The Seamless Checkout is truly frictionless and simplified through a single RaaS (Registration as a Service) API.

LESSON SUMMARY

You should now be able to:

- Explain the basics of the SAP Customer Data Cloud (CDC) platform and API

UNIT 2 Console

UNIT OBJECTIVES

- Use the SAP Customer Data Cloud console

Unit 2
Lesson 1

Understanding the SAP Customer Data Cloud console

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use the SAP Customer Data Cloud console

Console

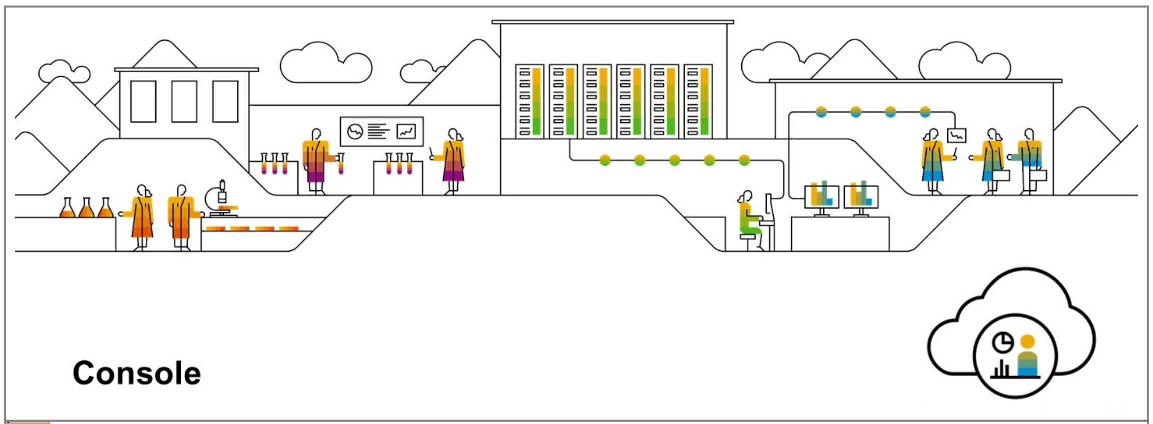


Figure 12: Console

- ***Partner*** and ***Site*** selector
- Cloud Web Tool for creating and configuring ***Digital Properties*** (*Sites*)
- Deal with currently selected site *configuration* (***Settings***, ***Policies***, ***Permissions***, ***Screen-Sets***, and more)
- ***Reporting*** capabilities and ***Identity Access*** tool
- Invite / Manage ***Administrators***, ***Applications*** and ***Permissions Groups***
- Access to **Consent Vault** and **Audit log**
- SAP Fiori Look & Feel

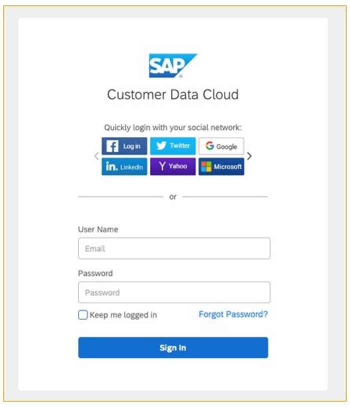


Figure 13: SAP Customer Data Cloud Console

The first step of Customer Data Cloud implementation is to create your site domain, adding its name to the sites table in the Dashboard

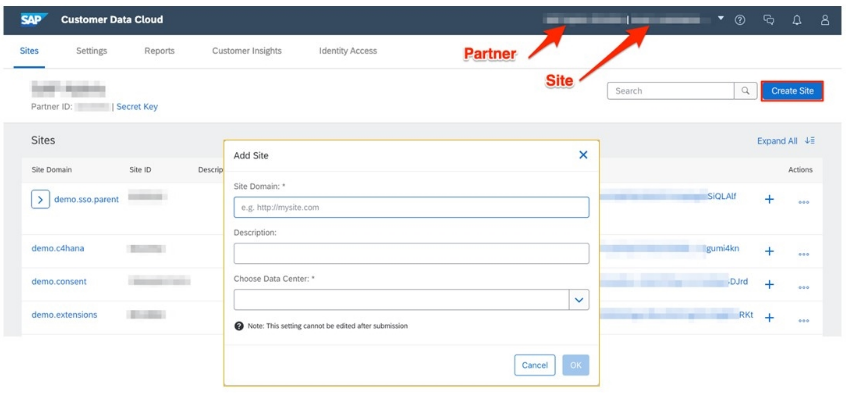


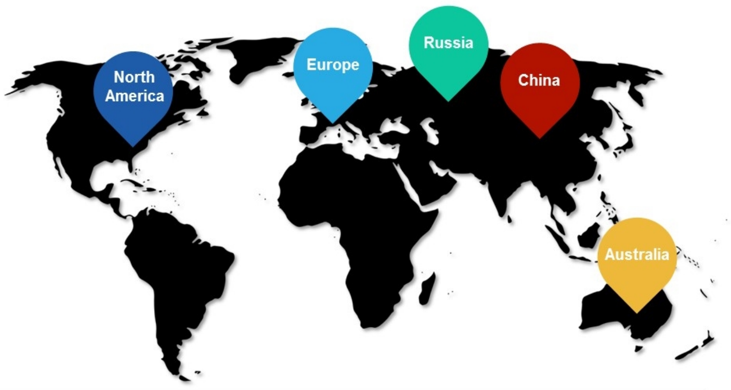
Figure 14: Site Setup

- For example, if you use three environments (Development, Staging and Production), you will need separate site definitions for each environment (3 unique API keys).
- The data center entry cannot be changed once the site has been created and indicates the location of the server holding your user data. Selecting the correct data center is not trivial and is dependent on your site's location. To verify a site's data center location, see

https://developers.gigya.com/display/GD/Finding+Your+Data+Center .

Customer Data Cloud offers 5 primary data centers, each with full data residency:

- North America
- Europe
- Australia
- Russia
- China



Sensitive data encrypted (in transit and at rest)

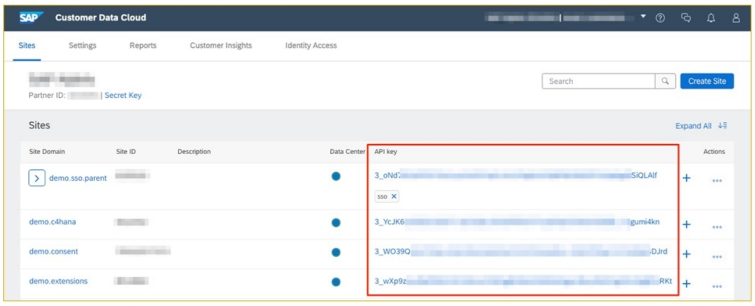
Robust and detailed audit logging

Figure 15: Data Residency

The data center entry indicates the location of the server holding your user data.

It cannot be changed once the site has been created.

- This API key will be referenced in every page in which Customer Data Cloud plugins or API calls are integrated
- The key is used to identify the calling site and assess the permissions available for that site



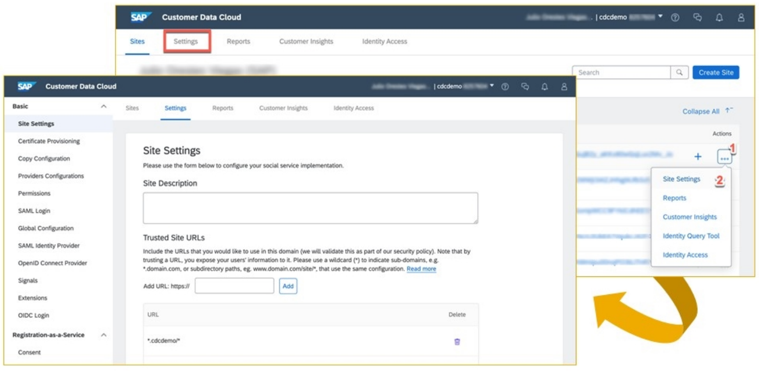
Sites

Site Domain	Site ID	Description	Data Center	API key	Actions
demo.sso.parent			●	3_oNd SiQLAlf sso	+ ...
demo.c4hana			●	3_YcJK6 gumi4kn	+ ...
demo.consent			●	3_WO39Q DJrd	+ ...
demo.extensions			●	3_wXp9z RKt	+ ...

Figure 16: API Key

Production environment API keys should not be used for testing. Use separate site definitions and API keys for test and production sites.

- Use Site Settings to ***configure*** your SAP CDC ***Identity***, ***Consent*** and ***Profile*** services implementation



Site Settings

Please use the form below to configure your social service implementation.

Site Description

Trusted Site URLs

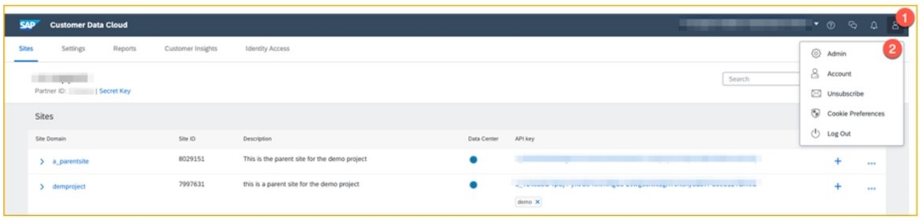
Include the URLs that you would like to use in this domain (we will validate this as part of our security policy). Note that by trusting a URL, you expose your users' information to it. Please use a wildcard (*) to indicate sub-domains, e.g. *.domain.com, or subdirectory paths, eg. www.domain.com/site/*, that use the same configuration. Read more

Add URL: https:// Add

URL	Delete
.cdcdemo/	

Figure 17: Site Settings

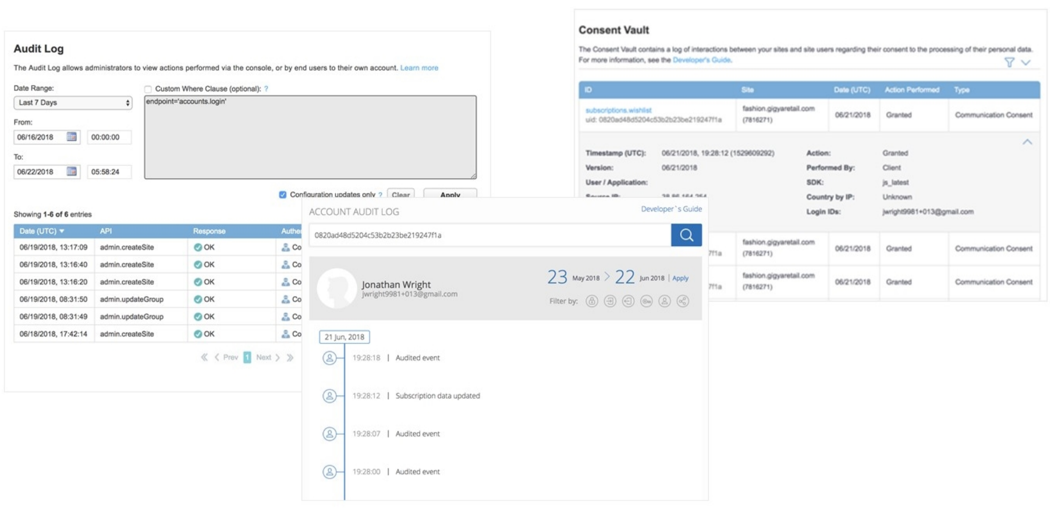
- Customer Data Cloud provides a general **administration** framework for setting and enforcing **user-level** and **application-level** permissions



Sites

Site Domain	Site ID	Description	Data Center	API key
a_parentsite	8029151	This is the parent site for the demo project	●	
demproject	7997631	this is a parent site for the demo project	●	demo

Figure 18: Console Administration



Audit Log

The Audit Log allows administrators to view actions performed via the console, or by end users to their own account. Learn more

Date Range: Last 7 Days

From: 06/16/2018 00:00:00

To: 06/22/2018 05:58:24

Custom Where Clause (optional): ?

endpoint='accounts.login'

Configuration updates only ? Clear Apply

Showing **1-6 of 6** entries

Date (UTC)	API	Response	Authe
06/19/2018, 13:17:09	admin.createSite	OK	Co
06/19/2018, 13:16:40	admin.createSite	OK	Co
06/19/2018, 13:16:20	admin.createSite	OK	Co
06/19/2018, 08:31:50	admin.updateGroup	OK	Co
06/19/2018, 08:31:49	admin.updateGroup	OK	Co
06/18/2018, 17:42:14	admin.createSite	OK	Co

ACCOUNT AUDIT LOG Developer`s Guide

0820ad48d5204c53b2b23be219247f1a

Jonathan Wright
jwright9981+013@gmail.com

23 May 2018 > 22 Jun 2018 | Apply

Filter by:

21 Jun, 2018

19:28:18 | Audited event

19:28:12 | Subscription data updated

19:28:07 | Audited event

19:28:00 | Audited event

Consent Vault

The Consent Vault contains a log of interactions between your sites and site users regarding their consent to the processing of their personal data. For more information, see the Developer's Guide.

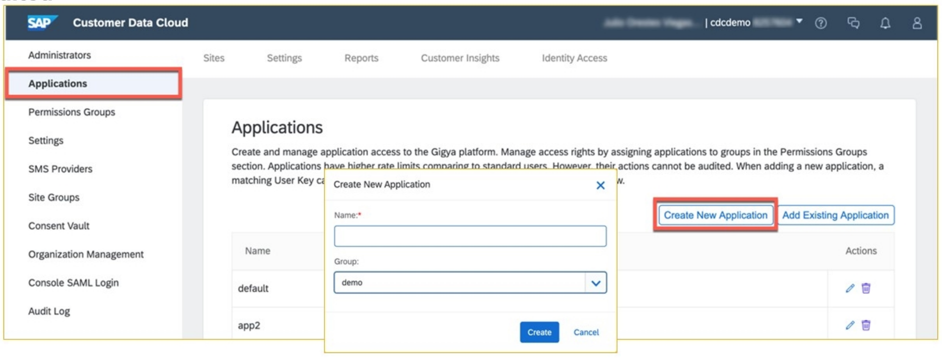
ID	Site	Date (UTC)	Action Performed	Type
subscriptions.wishlist uid: 0820ad48d5204c53b2b23be219247f1a	fashion.gigyaretail.com (7816271)	06/21/2018	Granted	Communication Consent

Timestamp (UTC): 06/21/2018, 19:28:12 (1529609292)
Version: 06/21/2018
User / Application:
Action: Granted
Performed By: Client
SDK: js_latest
Country by IP: Unknown
Login IDs: jwright9981+013@gmail.com

ID	Site	Date (UTC)	Action Performed	Type
7f1a	fashion.gigyaretail.com (7816271)	06/21/2018	Granted	Communication Consent
7f1a	fashion.gigyaretail.com (7816271)	06/21/2018	Granted	Communication Consent

Figure 19: Auditing Capabilities

- **Application Keys** are credentials that are given to third-party applications to enable them to access the Customer Data Cloud platform and make system calls
- Applications may have ***higher rate limits*** than administrator users, but their actions are ***not audited***



Applications

Create and manage application access to the Gigya platform. Manage access rights by assigning applications to groups in the Permissions Groups section. Applications have higher rate limits comparing to standard users. However, their actions cannot be audited. When adding a new application, a matching User Key ca[illegible]w.

Create New Application

Name:*

Group: demo

Create Cancel

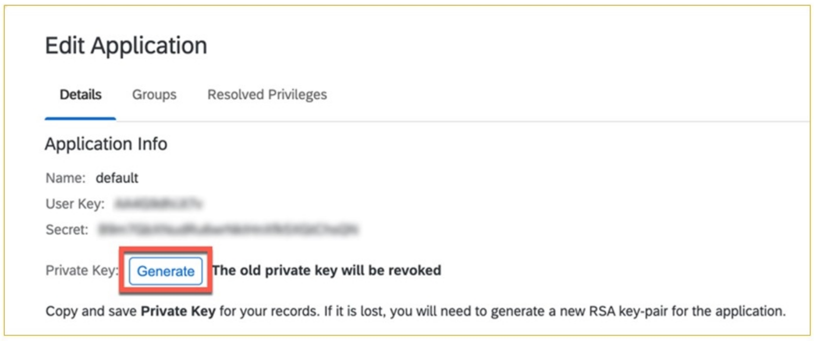
Create New Application Add Existing Application

Name	Actions
default	
app2	

Figure 20: Manage Applications

https://developers.gigya.com/display/GD/Acceptable+Use+Policy

- If the app keys get lost or leaked, they can be regenerated



Edit Application

Details Groups Resolved Privileges

Application Info

Name: default

User Key: [redacted]

Secret: [redacted]

Private Key: Generate **The old private key will be revoked**

Copy and save **Private Key** for your records. If it is lost, you will need to generate a new RSA key-pair for the application.

Figure 21: Regenerate application key

- Create and manage **administrators** with access to the CDC Console
- Manage access rights by assigning users to groups in the **Permissions Groups** section

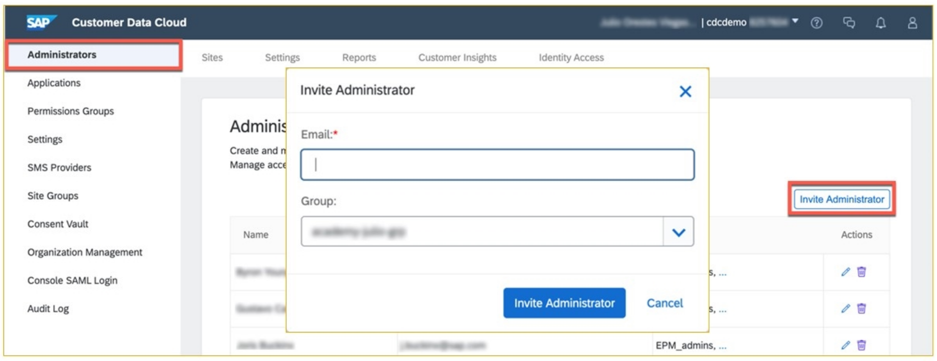


Figure 22: Invite Administrators

The Permission Group system enables you to create users (user keys), grant them permissions, and evaluate their permissions upon incoming requests. The permissions determine which API methods the user can call, what parameters the user can pass, what the valid values for these parameters are and what types of logical operations are allowed. In addition, permissions are scoped to certain partners or sites.

- User keys are used to grant individual permissions to certain users on certain sites

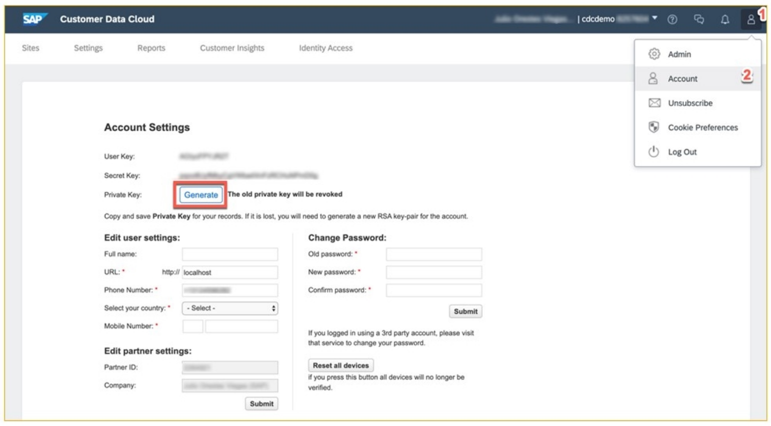


Figure 23: Account Settings

^xUser keys are more secure than giving all users the partner secret key, which grants full permission to all data and actions on the API key, including the ability to delete user data. In addition, actions taken using the user key are tracked for auditing purposes.

- Partner ID is a unique identifier to identify your Customer Data Cloud instance.
- Secret Key may be used to generate and check Cryptographic Signatures to verify the authenticity of Customer Data Cloud processes and prevent fraud

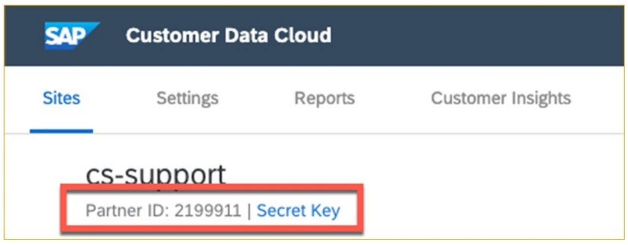
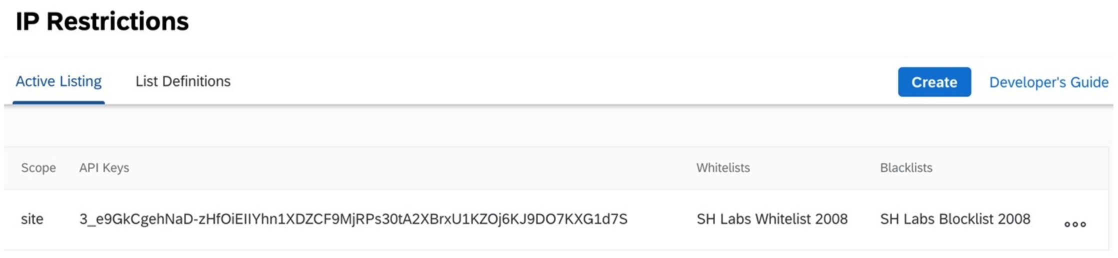


Figure 24: Partner ID and Secret Key

Caution 1: Never use the partner's secret key in your configuration or developments!!!

Caution 2: Never give the secret key! Think of it as your root password for your entire partner account!

- Control the IP addresses that can access Customer Data APIs
- Configure either whitelists, blacklists, or both in Customer Data Console
- Configured at the Partner level or for specific API keys in the Active Listings configuration
- Create List Definitions & Active Listings on IP Restrictions page



Scope	API Keys	Whitelists	Blacklists
site	3_e9GkCgehNaD-zHfOiEIlYhn1XDZCF9MjRPs30tA2XBrxU1KZOj6KJ9DO7KXG1d7S	SH Labs Whitelist 2008	SH Labs Blocklist 2008

Figure 25: IP Restrictions

https://developers.gigya.com/display/GD/Console+Administration#ConsoleAdministration-IPRestrictions

- Let’s explore the SAP CDC Console together!

Figure 26: Administrator - Walkthrough

Internet Explorer is not supported anymore.

Refer this video for more details

https://enable.cx.sap.com/playlist/dedicated/95027061/0_cshu70of/1_l6iylh9e

Site Setup

https://enable.cx.sap.com/media/1_cx0inbw5

Figure 27: Exercise 1

LESSON SUMMARY

You should now be able to:

- Use the SAP Customer Data Cloud console

UNIT 3 Schema

UNIT OBJECTIVES

- Describe the SAP Customer Data Cloud data schema

Understanding the Schema

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Describe the SAP Customer Data Cloud data schema

Data Schema

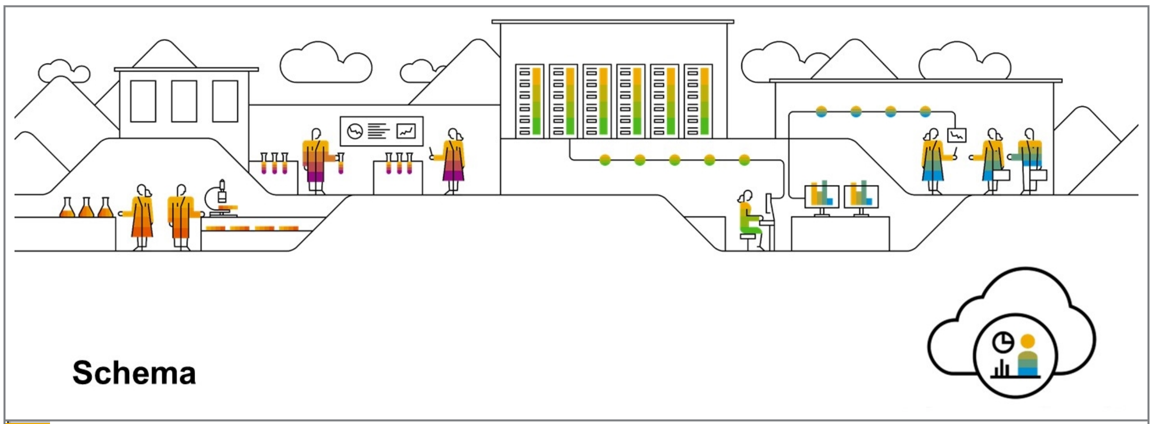
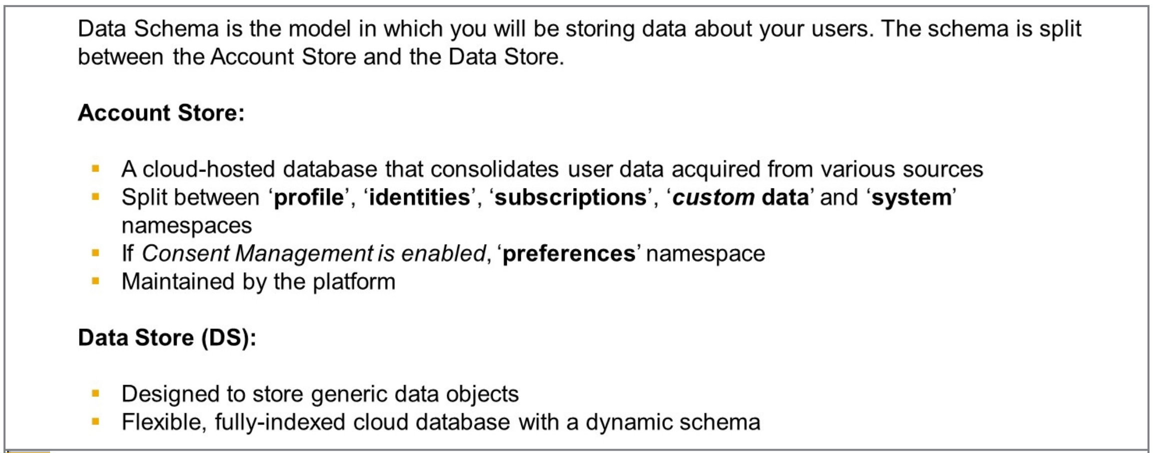


Figure 28: Data Schema



Data Schema is the model in which you will be storing data about your users. The schema is split between the Account Store and the Data Store.

Account Store:

- A cloud-hosted database that consolidates user data acquired from various sources
- Split between '**profile**', '**identities**', '**subscriptions**', '***custom* data**' and '**system**' namespaces
- If *Consent Management is enabled*, '**preferences**' namespace
- Maintained by the platform

Data Store (DS):

- Designed to store generic data objects
- Flexible, fully-indexed cloud database with a dynamic schema

Figure 29: What is the Schema?

- System data fields are not displayed in the schema, but they are returned with the payload of the login request or response.
- Identity Store is the main store (Login, Accounts API, etc.)
- Data Store (DS) is an API Database (JSON) which can be customer, site, or project-specific. It's a cloud database, and the data is merged automatically with Identity Data.

Account and Data Store

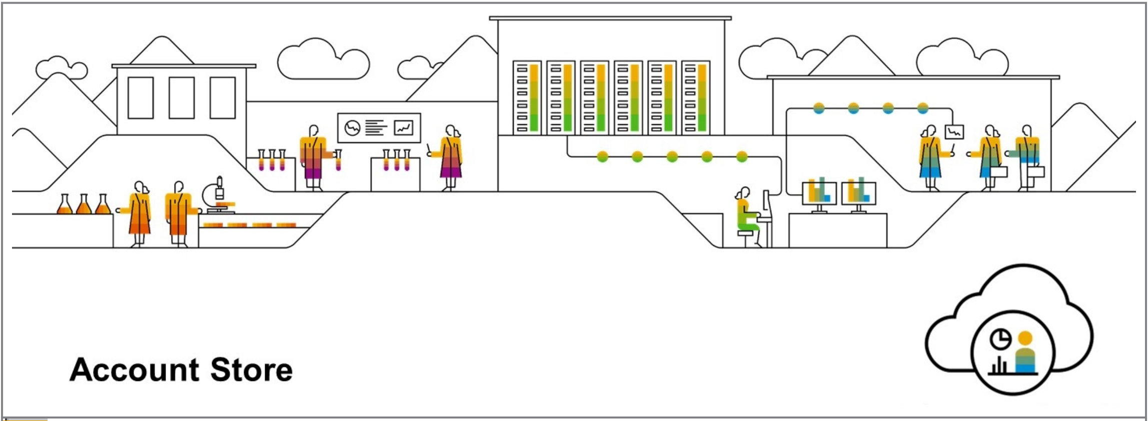


Figure 30: Account and Data Store

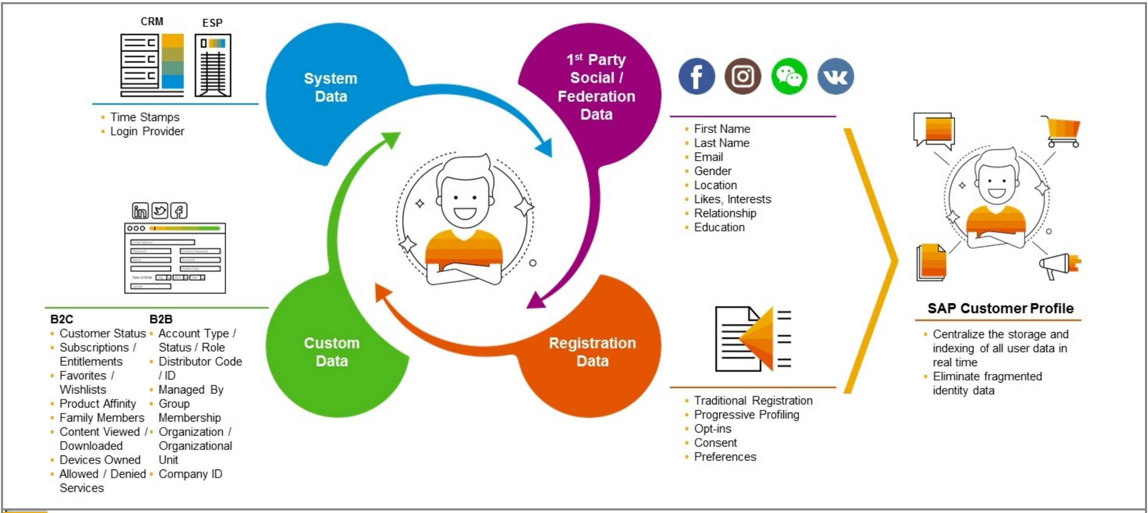
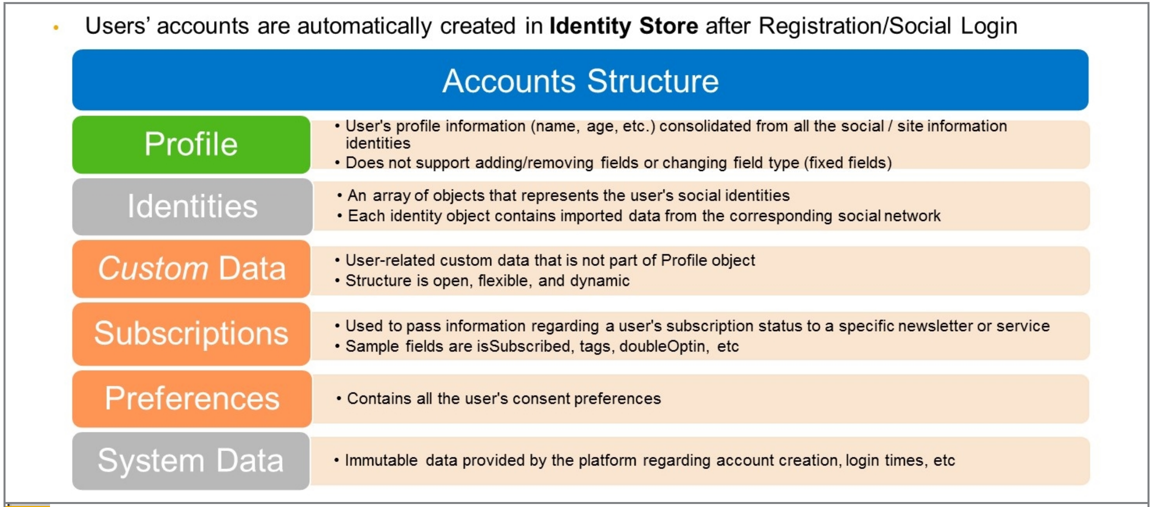


Figure 31: SAP Customer Profile - Flexible Account Structure



- Users' accounts are automatically created in **Identity Store** after Registration/Social Login

Accounts Structure	
Profile	• User's profile information (name, age, etc.) consolidated from all the social / site information identities • Does not support adding/removing fields or changing field type (fixed fields)
Identities	• An array of objects that represents the user's social identities • Each identity object contains imported data from the corresponding social network
Custom Data	• User-related custom data that is not part of Profile object • Structure is open, flexible, and dynamic
Subscriptions	• Used to pass information regarding a user's subscription status to a specific newsletter or service • Sample fields are isSubscribed, tags, doubleOptin, etc
Preferences	• Contains all the user's consent preferences
System Data	• Immutable data provided by the platform regarding account creation, login times, etc

Figure 32: The User Account

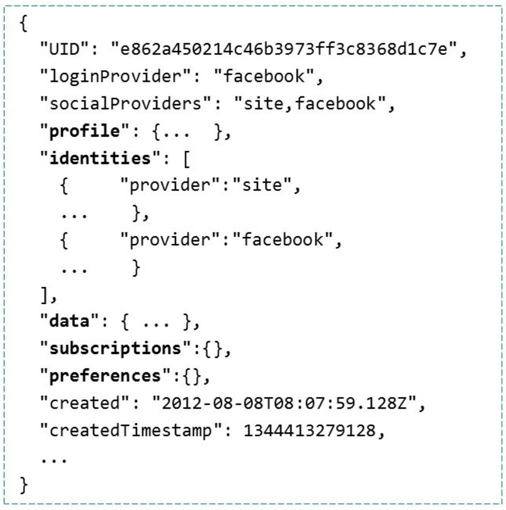
Refer to this link for more details

https://developers.gigya.com/display/GD/Profile+Management+-+IDS

Social Sync - The user's social network profile data (which is available after Registration/ Social Login) is automatically imported, stored in Profile Management, and made available for searching. In addition, whenever your Facebook users update their profiles on Facebook, the changes will sync automatically to Profile Management. CDC automatically subscribes to the following Facebook fields: birthday, books, education, email, first_name, hometown, last_name, likes, link, locale, location, movies, music, name, relationship_status, religion, verified, timezone and work. Note that CDC complies with the social networks' platform policies.

Dynamic Schema - The storage is built with a dynamic schema that can seamlessly process massive amounts of user data in an optimized way. Having a dynamic schema means that you may store any custom data you have with no constraints on its structure. You don't have to know in advance how your custom data is going to look, and it doesn't have to look the same for all objects. There is no need to go through schema creation or modification when the data structure changes.

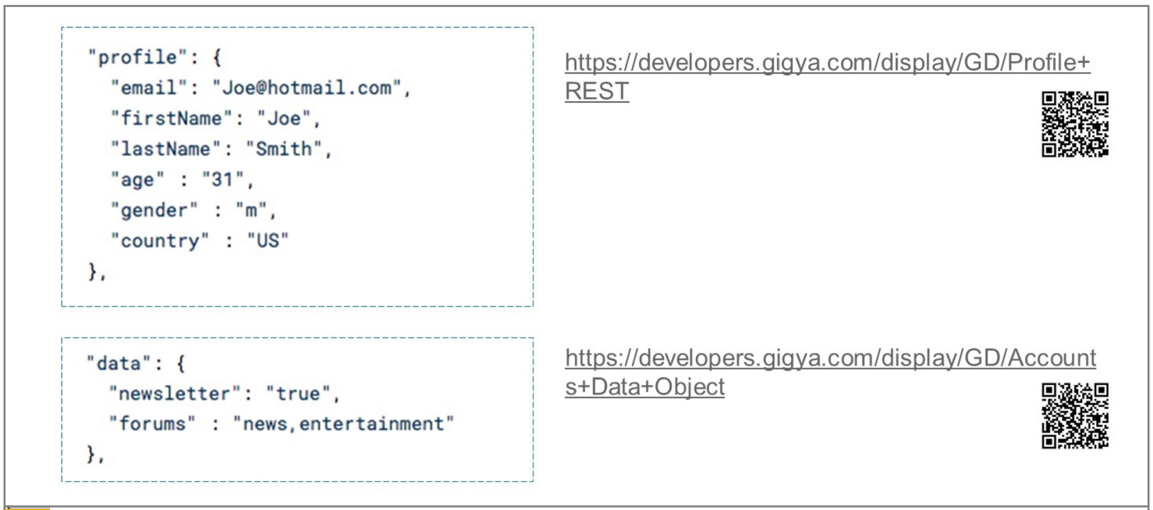
- PROFILE consists of first name, last name, email, and up to 35 fields - it's fixed - you cannot change it, except whether it is mandatory or not. Fixed fields are used for integrations with FB, Google, etc.
- DATA is a custom layer for data (change names, types, address, postcode or zip code, etc). There is no way at the moment to create a handler between profile and data
- SUBSCRIPTIONS consists of anything related to user subscriptions but not with user consent or user preferences: I consent with terms but I do not subscribe to this email letter.... (opt-in)
- When you login you get returned only data that has a value (is not null)
- Best practice: turn off dynamic schema



```
{
  "UID": "e862a450214c46b3973ff3c8368d1c7e",
  "loginProvider": "facebook",
  "socialProviders": "site,facebook",
  "profile": {...  },
  "identities": [
    {     "provider":"site",
    ...     },
    {     "provider":"facebook",
    ...     }
  ],
  "data": { ... },
  "subscriptions":{},
  "preferences":{},
  "created": "2012-08-08T08:07:59.128Z",
  "createdTimestamp": 1344413279128,
  ...
}
```

https://developers.gigya.com/display/GD/Profile+Management+-+IDS#ProfileManagement-IDS-AccountStructure

Figure 33: A sample Account in JSON object representation



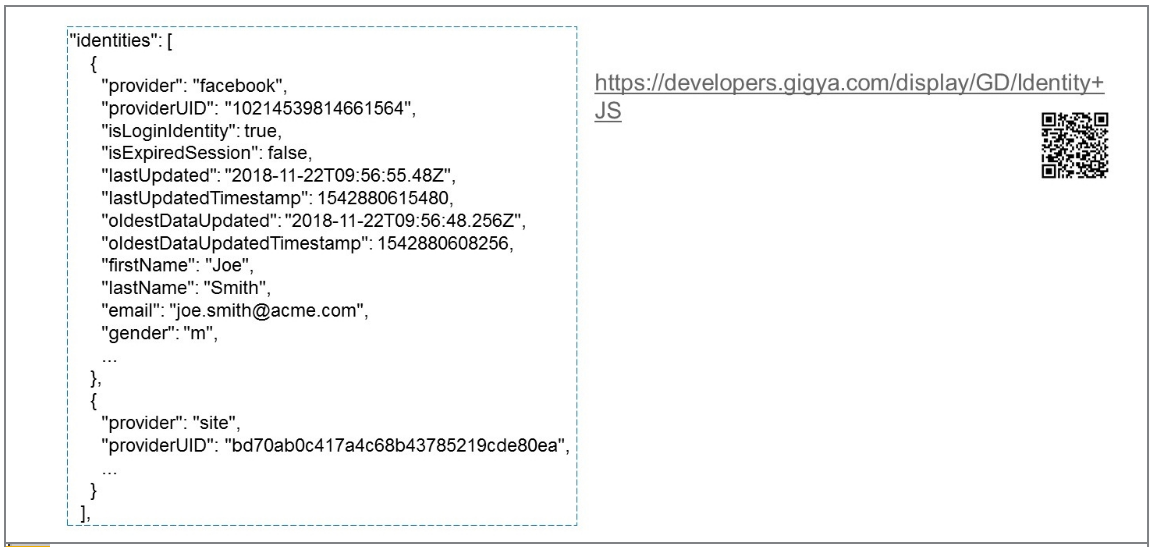
```
"profile": {
  "email": "Joe@hotmail.com",
  "firstName": "Joe",
  "lastName": "Smith",
  "age" : "31",
  "gender" : "m",
  "country" : "US"
},
```

https://developers.gigya.com/display/GD/Profile+REST

```
"data": {
  "newsletter": "true",
  "forums" : "news,entertainment"
},
```

https://developers.gigya.com/display/GD/Accounts+Data+Object

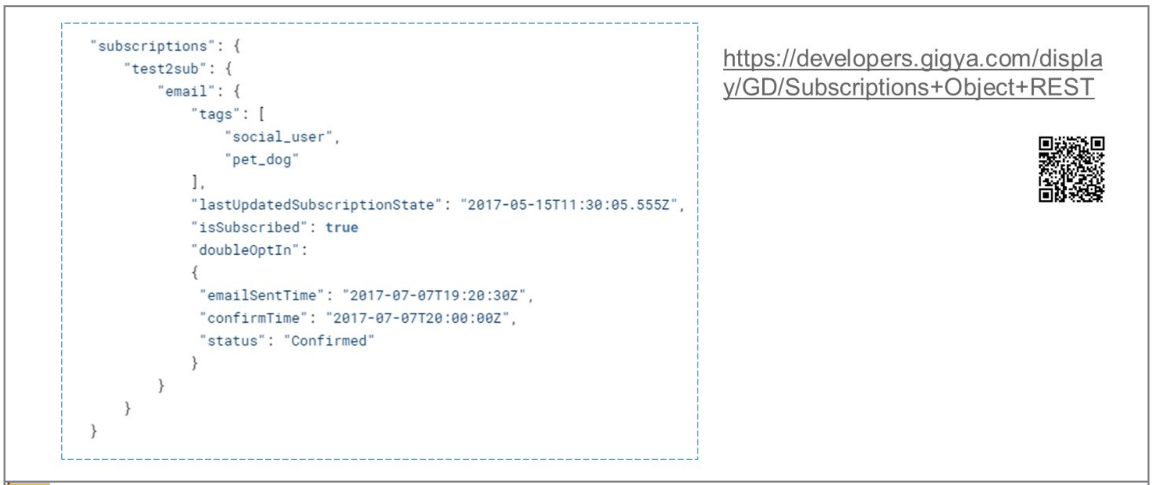
Figure 34: A Sample Account in JSON Object Representation - Profile and Custom Data



```
"identities": [
  {
    "provider": "facebook",
    "providerUID": "10214539814661564",
    "isLoginIdentity": true,
    "isExpiredSession": false,
    "lastUpdated": "2018-11-22T09:56:55.48Z",
    "lastUpdatedTimestamp": 1542880615480,
    "oldestDataUpdated": "2018-11-22T09:56:48.256Z",
    "oldestDataUpdatedTimestamp": 1542880608256,
    "firstName": "Joe",
    "lastName": "Smith",
    "email": "joe.smith@acme.com",
    "gender": "m",
    ...
  },
  {
    "provider": "site",
    "providerUID": "bd70ab0c417a4c68b43785219cde80ea",
    ...
  }
],
```

https://developers.gigya.com/display/GD/Identity+JS

Figure 35: A Sample Account in JSON Object Representation - Identities

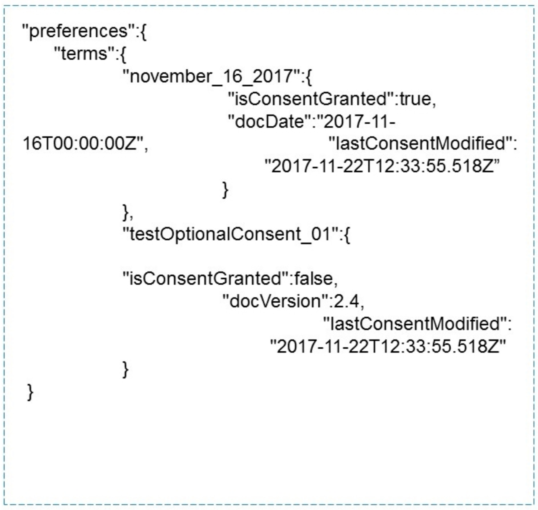


```
"subscriptions": {
    "test2sub": {
        "email": {
            "tags": [
                "social_user",
                "pet_dog"
            ],
            "lastUpdatedSubscriptionState": "2017-05-15T11:30:05.555Z",
            "isSubscribed": true
            "doubleOptIn":
            {
             "emailSentTime": "2017-07-07T19:20:30Z",
             "confirmTime": "2017-07-07T20:00:00Z",
             "status": "Confirmed"
            }
        }
    }
}
```

https://developers.gigya.com/display/GD/Subscriptions+Object+REST

Figure 36: A Sample Subscription in JSON Object Representation - Subscriptions

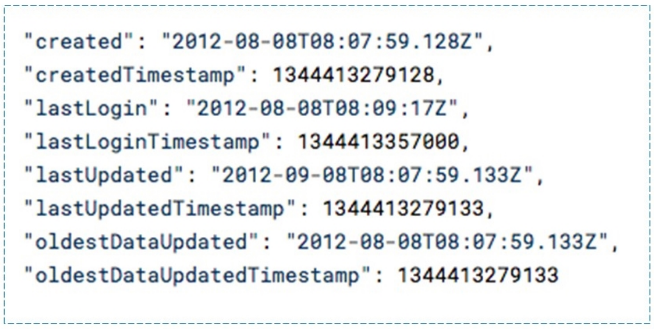
https://developers.gigya.com/display/GD/Subscriptions+Object+REST



```
"preferences":{
     "terms":{
               "november_16_2017":{
                              "isConsentGranted":true,
                              "docDate":"2017-11-
16T00:00:00Z",                          "lastConsentModified":
                                   "2017-11-22T12:33:55.518Z"
                              }
               },
               "testOptionalConsent_01":{

               "isConsentGranted":false,
                              "docVersion":2.4,
                                        "lastConsentModified":
                                   "2017-11-22T12:33:55.518Z"
               }
}
```

https://developers.gigya.com/display/GD/Preferences+Object+REST

Figure 37: A Sample Account in JSON Object Representation - Preferences



```
"created": "2012-08-08T08:07:59.128Z",
"createdTimestamp": 1344413279128,
"lastLogin": "2012-08-08T08:09:17Z",
"lastLoginTimestamp": 1344413357000,
"lastUpdated": "2012-09-08T08:07:59.133Z",
"lastUpdatedTimestamp": 1344413279133,
"oldestDataUpdated": "2012-08-08T08:07:59.133Z",
"oldestDataUpdatedTimestamp": 1344413279133
```

Figure 38: A Sample Account in JSON Object Representation - System Data

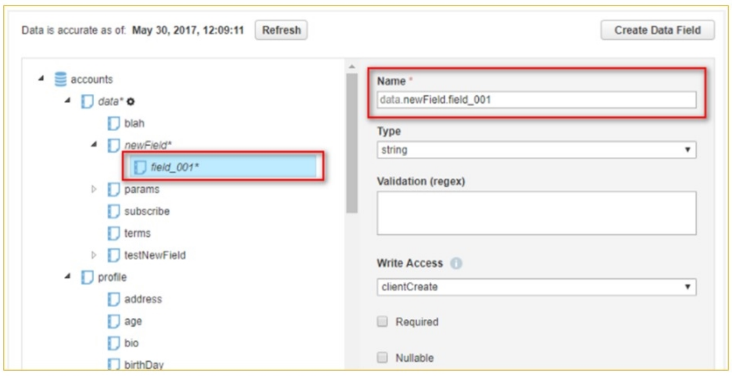
- The data that can be returned from social providers upon a login/social registration is mapped to the profile object using **Social Sync**
- CDC uses Social Sync to monitor and track Facebook Webhooks. Social Sync enables your user's account data to be automatically updated whenever your Facebook users **update** their profiles on Facebook; these changes will sync automatically to their existing accounts in CDC.
- When a user **deletes** your site's Facebook application from their Facebook Apps, CDC automatically deletes all the Facebook data kept in the user's account except for the following fields: snuid, firstName, lastName, nickname, profilePhoto, and gender. This happens automatically and no action is required by your site's admin.

Figure 39: What is SocialSync?

https://developers.gigya.com/display/GD/Identity+Compliance#IdentityCompliance-IdentityCompliance:FacebookData

As the profile object is mapped with the data that can be returned from social providers, Profile object is pre-defined and cannot be modified.

- Interactively edit schema in the Customer Data Cloud Console
- Supports up to 1000 custom fields
- Available to console users that have the necessary Console permissions to use at least one of the following APIs:
  - accounts.getSchema
  - accounts.setSchema


Data is accurate as of: May 30, 2017, 12:09:11

Name *: data.newField.field_001

Type: string

Validation (regex):

Write Access: clientCreate

☐ Required

☐ Nullable

Figure 40: Schema Editor

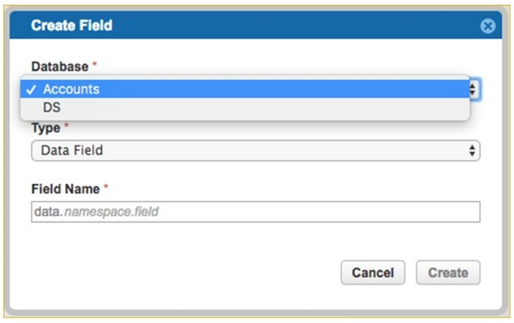
Overview:

- Interactively edit schema in the Customer Data Cloud Console
- Support up to 1000 additional custom fields beyond the default schemas
- New fields can be added to existing site's schema
- This utility is only available to Console users that have the necessary Console permissions

Benefits:

- Clients can now easily view their existing schemas
- Clients can quickly update their schemas themselves: self service
- Only authorized individuals are granted access to Schema Editor for extra security

Create new field

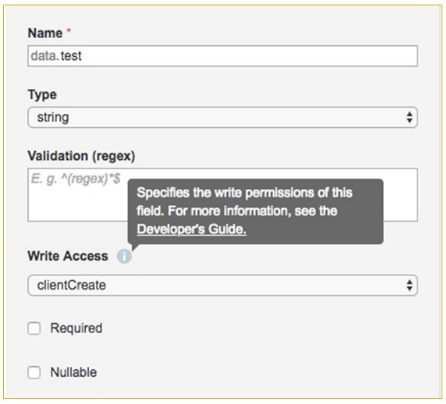

Create Field

Database *: ✓ Accounts / DS

Type *: Data Field

Field Name *: data.namespace.field

Configure the properties of the field


Name *: data.test

Type: string

Validation (regex): E. g. ^(regex)*$

Specifies the write permissions of this field. For more information, see the Developer's Guide.

Write Access: clientCreate

☐ Required

☐ Nullable

Figure 41: Define Schema using Console

writeAccess - Specifies whether to allow unsigned requests to write into this field. This property applies when using the accounts.setAccountInfo method or when setting fields through the usage of a Screen-Set.

The supported values are:

"serverOnly" (default) - Only signed requests coming from the server are allowed.

"clientCreate" - Unsigned requests coming from the client are allowed to write into this field, if it was not previously set.

"clientModify" - Unsigned requests coming from the client are allowed to write into this field and modify existing values.

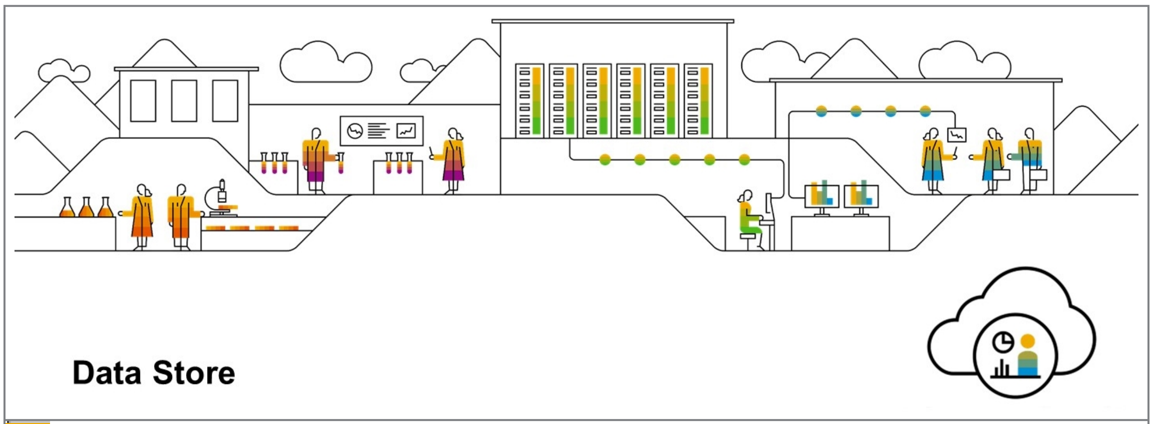
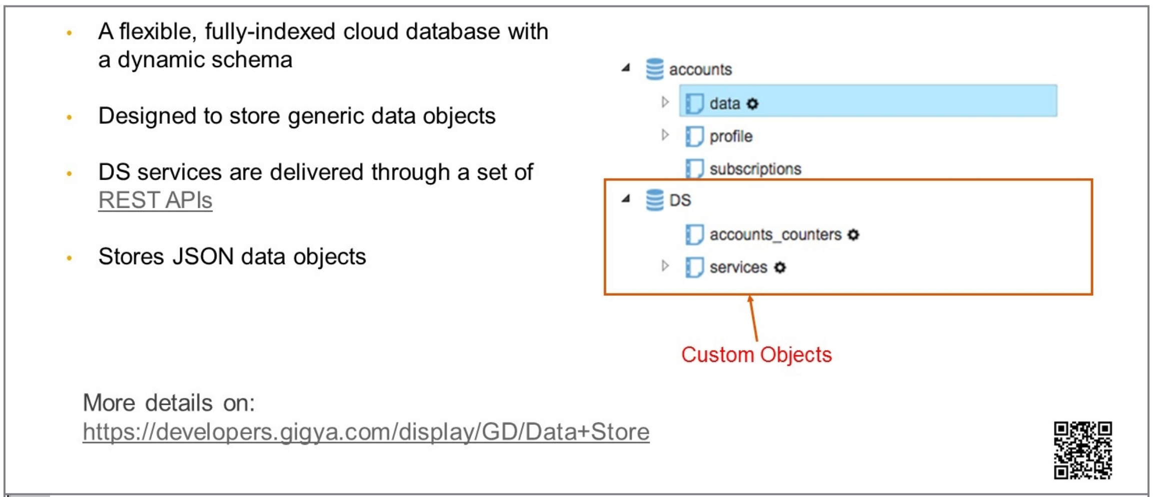


Figure 42: Account and Data Store



- A flexible, fully-indexed cloud database with a dynamic schema
- Designed to store generic data objects
- DS services are delivered through a set of REST APIs
- Stores JSON data objects

accounts
data
profile
subscriptions
DS
accounts_counters
services
Custom Objects

More details on:
https://developers.gigya.com/display/GD/Data+Store

Figure 43: Data Store

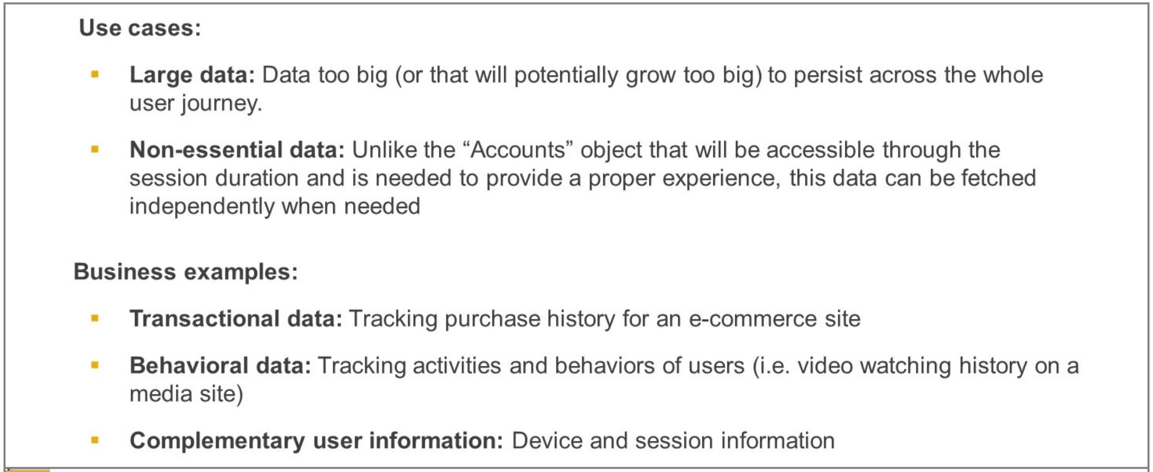
The Data Store is a premium platform that requires separate activation.

Limitation of 512K per JSON object.

Major Features

- Dynamic Schema - Meaning you don't have to know in advance how your data is going to look and it doesn't have to look the same for all objects. You may store any data you have with no constrains on its structure. There is no need to go through schema creation or modification when the data structure changes.
- Fully Indexed - Meaning that no matter what data you send you can always search based on any combination of stored fields in that data. You don't have to know what your queries are going to be or create the appropriate index in advance. An example of such a search: get the email addresses of all U.S. users who graduated from any university between 2000-2005.

https://developers.gigya.com/display/GD/Data+Store



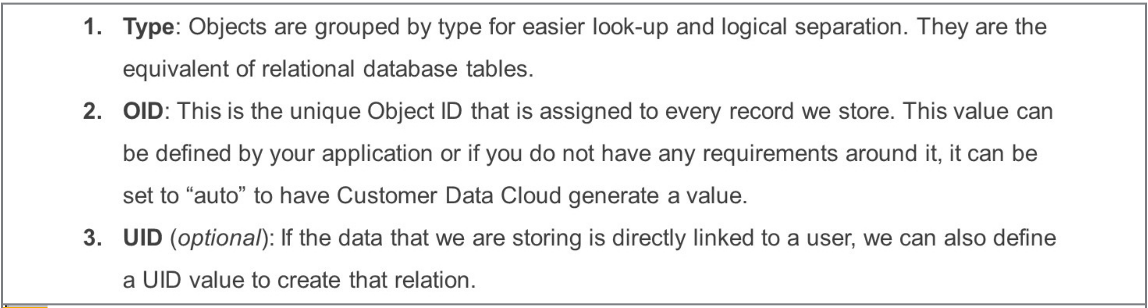
Use cases:

- **Large data:** Data too big (or that will potentially grow too big) to persist across the whole user journey.
- **Non-essential data:** Unlike the "Accounts" object that will be accessible through the session duration and is needed to provide a proper experience, this data can be fetched independently when needed

Business examples:

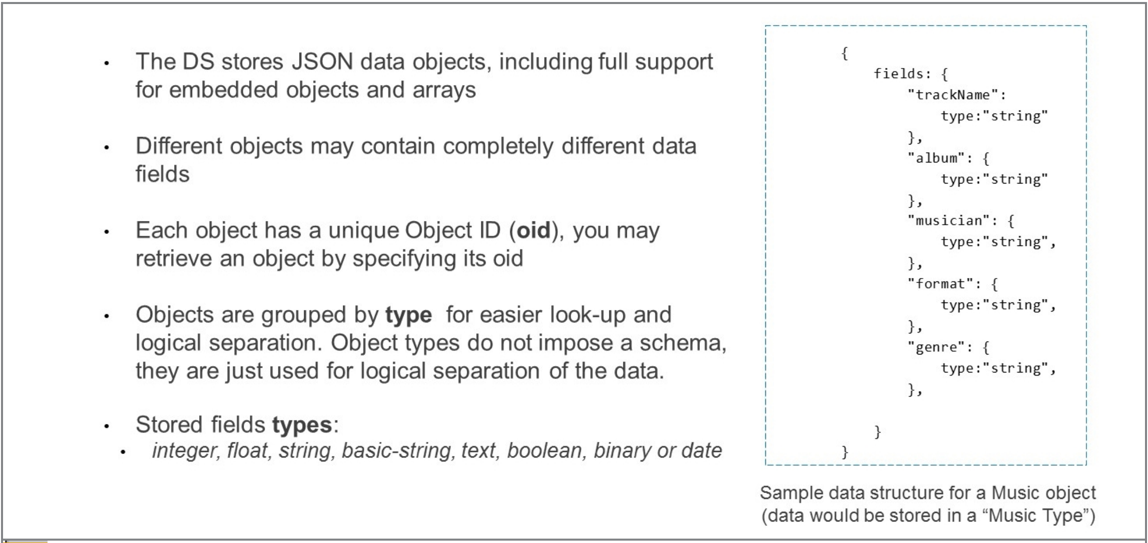
- **Transactional data:** Tracking purchase history for an e-commerce site
- **Behavioral data:** Tracking activities and behaviors of users (i.e. video watching history on a media site)
- **Complementary user information:** Device and session information

Figure 44: Common DS Use Cases



1. **Type**: Objects are grouped by type for easier look-up and logical separation. They are the equivalent of relational database tables.
2. **OID**: This is the unique Object ID that is assigned to every record we store. This value can be defined by your application or if you do not have any requirements around it, it can be set to "auto" to have Customer Data Cloud generate a value.
3. **UID** (*optional*): If the data that we are storing is directly linked to a user, we can also define a UID value to create that relation.

Figure 45: Object Key fields

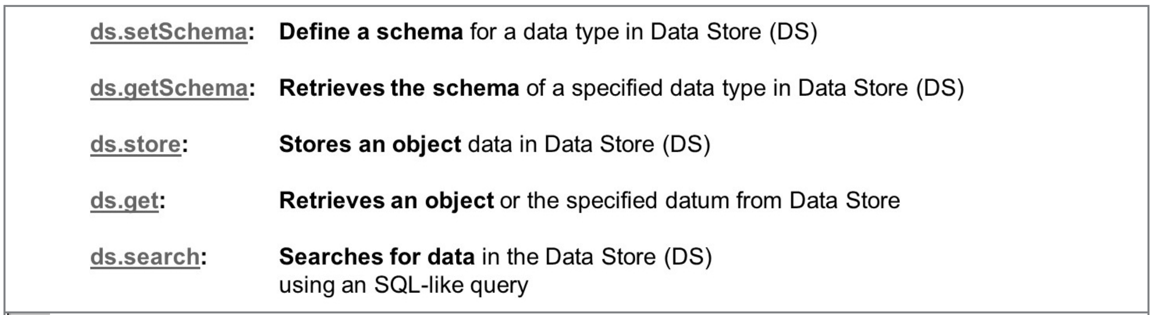


- The DS stores JSON data objects, including full support for embedded objects and arrays
- Different objects may contain completely different data fields
- Each object has a unique Object ID (**oid**), you may retrieve an object by specifying its oid
- Objects are grouped by **type** for easier look-up and logical separation. Object types do not impose a schema, they are just used for logical separation of the data.
- Stored fields **types**:
  - *integer, float, string, basic-string, text, boolean, binary or date*

```
{
    fields: {
        "trackName":
            type:"string"
        },
        "album": {
            type:"string"
        },
        "musician": {
            type:"string",
        },
        "format": {
            type:"string",
        },
        "genre": {
            type:"string",
        },

    }
}
```

Sample data structure for a Music object (data would be stored in a "Music Type")

Figure 46: Design your DS Objects



ds.setSchema:	**Define a schema** for a data type in Data Store (DS)
ds.getSchema:	**Retrieves the schema** of a specified data type in Data Store (DS)
ds.store:	**Stores an object** data in Data Store (DS)
ds.get:	**Retrieves an object** or the specified datum from Data Store
ds.search:	**Searches for data** in the Data Store (DS) using an SQL-like query

Figure 47: DS Key REST API Calls

https://developers.gigya.com/display/GD/ds.setSchema+REST

https://developers.gigya.com/display/GD/ds.getSchema+REST

https://developers.gigya.com/display/GD/ds.store+REST

https://developers.gigya.com/display/GD/ds.get+REST

https://developers.gigya.com/display/GD/ds.search+REST

Figure 48: Exercise 2

LESSON SUMMARY

You should now be able to:

- Describe the SAP Customer Data Cloud data schema

UNIT 4

Customer Identity

UNIT OBJECTIVES

- Describe the SAP Customer Identity
- Understand Lite Registration and Full Registration

Unit 4
Lesson 1

Understanding Registration as a Service

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Describe the SAP Customer Identity
- Understand Lite Registration and Full Registration

Customer Identity

Figure 49: Customer Identity

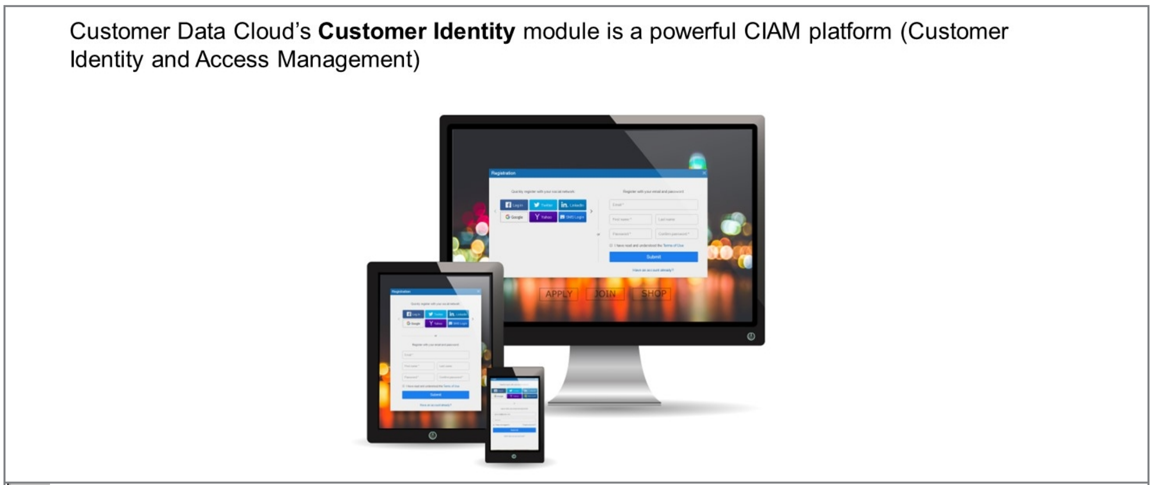

Customer Data Cloud's **Customer Identity** module is a powerful CIAM platform (Customer Identity and Access Management)

Figure 50: SAP Customer Identity

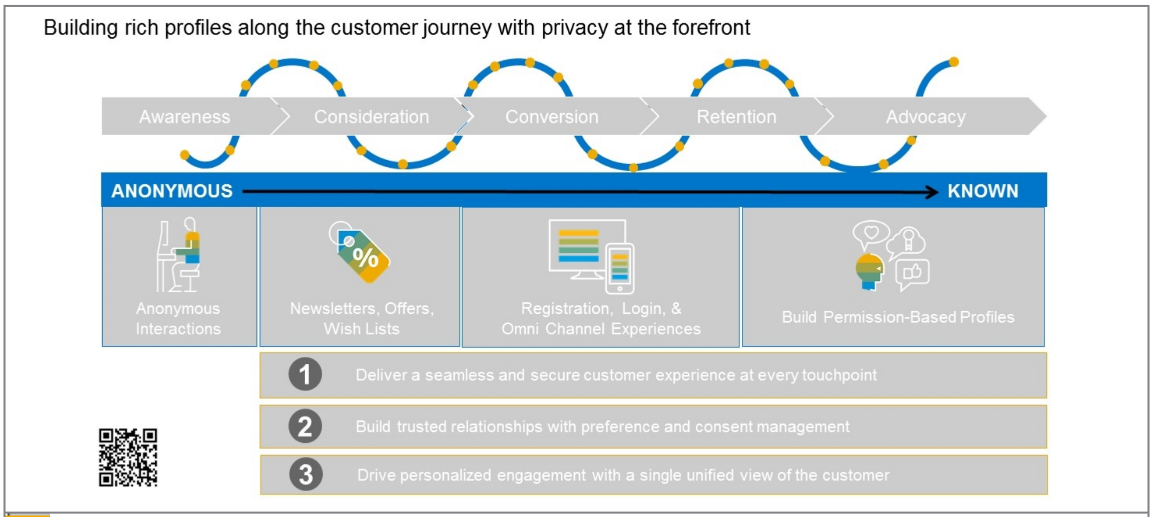

Building rich profiles along the customer journey with privacy at the forefront
Awareness
Consideration
Conversion
Retention
Advocacy
ANONYMOUS
KNOWN
Anonymous Interactions
Newsletters, Offers, Wish Lists
Registration, Login, & Omni Channel Experiences
Build Permission-Based Profiles
1 Deliver a seamless and secure customer experience at every touchpoint
2 Build trusted relationships with preference and consent management
3 Drive personalized engagement with a single unified view of the customer

Figure 51: Why you need to build profiles progressively

What we recommend is thinking about identity not from a binary concept (anonymous or known, unregistered to registered), but thinking about identity in a progressive way. The awareness phase is when the customer first visits your site. The customer is still anonymous, but there are things that you can learn about that consumer; whether it's through a mobile ad id, a cookie, etc.

Then the customer goes into the consideration phase. A full registration might be too much to ask for at this point, but maybe there is an opportunity to ask for what we often call a lite registration. Maybe that customer just wants to sign up for a newsletter on your website or is in a store and uses an email address to sign-up for a rewards program. The consideration phase is not getting a complete registration with a password and having to ask for that full authentication, but maybe just asking for an email address and starting to build that relationship with the customer.

The next step is the conversion stage, and at this stage is where you get that complete registration (as it's traditionally known). You get that full authentication of the customer. You might ask for things like the customer's address or various interests.

Retention is about building up the customer's profile; understanding the customer's preferences over time so you can drive personalization.

There are ways to do this. You may have heard of technologies like progressive profiling where over time, at the right times, you ask for additional information about the customer when you can provide that consumer value. Maybe it's about personalizing the website experience or personalizing the type of clothes or retail that you're showing to that consumer.

Finally, advocacy is when you can dynamically perform all these steps in the various phases on an ongoing basis. This is not a one-time thing. It's an ongoing process.

So many important attributes are stored about your customers. Many clients ask if there is a single centralized place to store key elements that enrich the single view of customer profile. Not only can we store that information we can also feed those centralized identities and profiles to downstream applications.

Refer following video for more details

https://enable.cx.sap.com/media/1_fyrbb6be/

Lite Registration

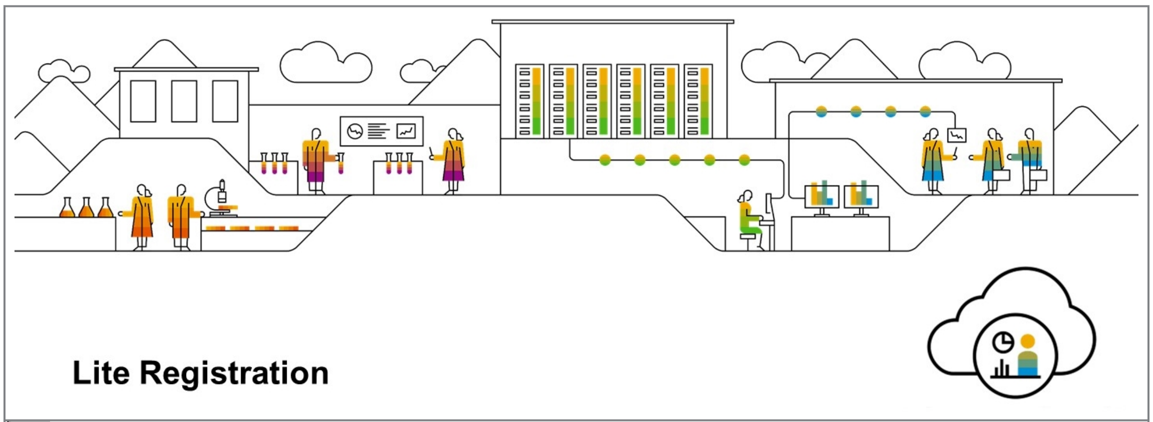
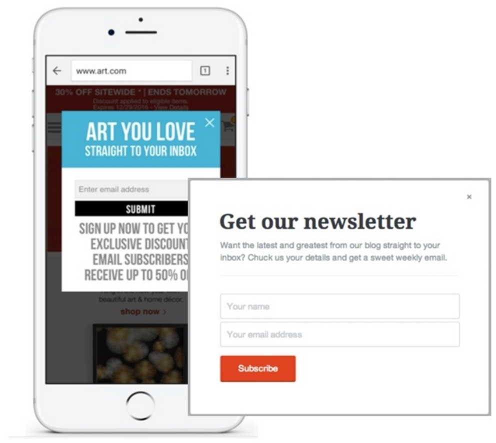


Figure 52: Lite Registration



- The first step of building a trust-based relationship between your site and the site users. The customer journey is started early
- Lite accounts are converted to Registered full accounts upon full registration
- After conversion, only the subscription values can be inherited by the full account

Figure 53: Lite Registration

Overview:

- Out-of-the-box lite registration screens
- Unregistered users can use an email to sign-up for content (e.g. promotional offers, newsletters) or to participate in a certain way (e.g. vote, unlock page)
- No password required
- Clear opt-in and opt-out flows for complying with privacy regulations
- Lite accounts are converted to registered accounts upon full registration
- Full support for exporting and importing subscriber data to third-party platforms, such as Email Service Providers (ESPs), either with IdentitySync (CDC's ETL service) or using accounts.search.

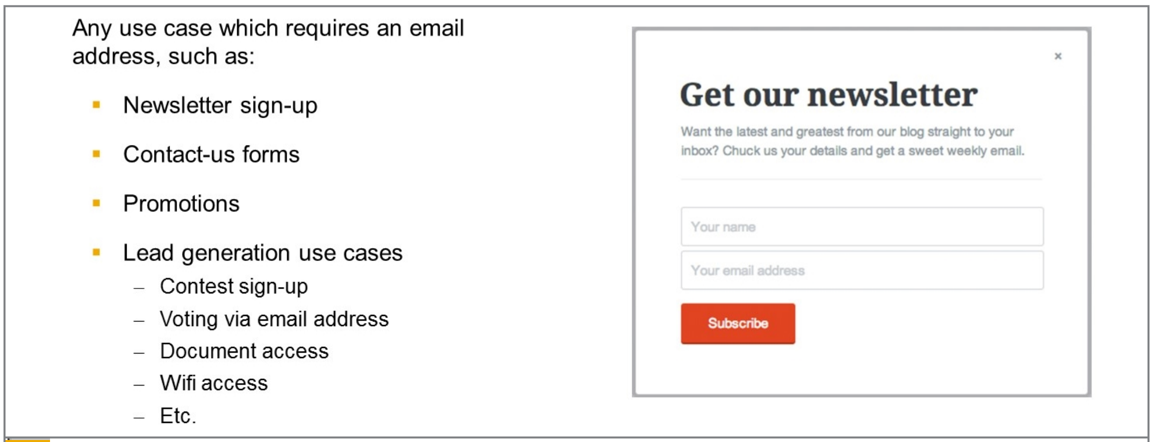
Benefits:

- Drive engagement based on value for information exchange
- Increase the number of known users

- Centralize opt-in management for registered and non-registered users

Refer to the following videos:

https://enable.cx.sap.com/media/1_g5m0638r



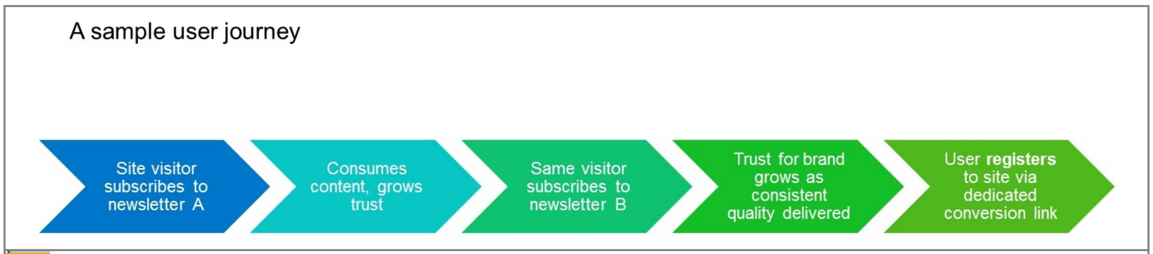
Any use case which requires an email address, such as:

- Newsletter sign-up
- Contact-us forms
- Promotions
- Lead generation use cases
  - Contest sign-up
  - Voting via email address
  - Document access
  - Wifi access
  - Etc.

×
Get our newsletter
Want the latest and greatest from our blog straight to your inbox? Chuck us your details and get a sweet weekly email.
Your name
Your email address
Subscribe

Figure 54: Supported Use Cases

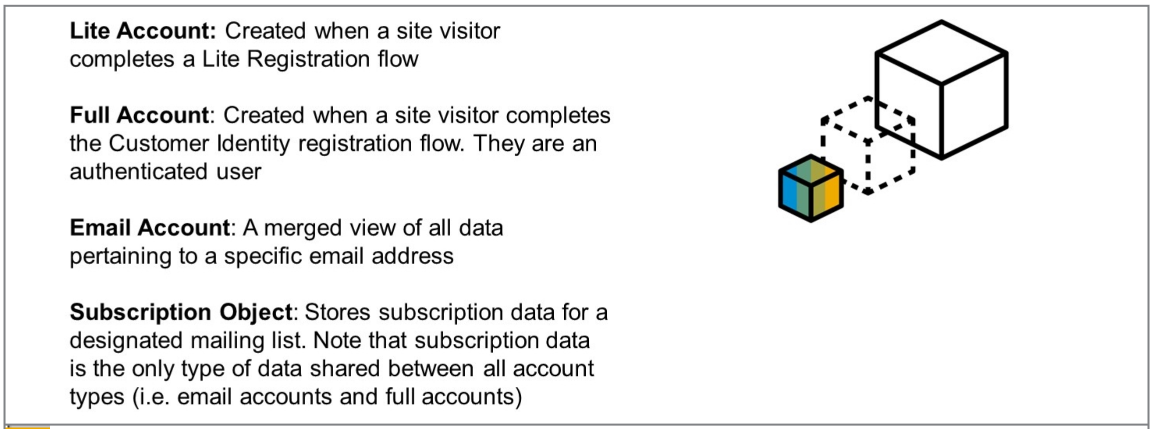
Email address is mandatory



A sample user journey

Site visitor subscribes to newsletter A
Consumes content, grows trust
Same visitor subscribes to newsletter B
Trust for brand grows as consistent quality delivered
User registers to site via dedicated conversion link

Figure 55: Lite Registration Sample Flow



Lite Account: Created when a site visitor completes a Lite Registration flow

Full Account: Created when a site visitor completes the Customer Identity registration flow. They are an authenticated user

Email Account: A merged view of all data pertaining to a specific email address

Subscription Object: Stores subscription data for a designated mailing list. Note that subscription data is the only type of data shared between all account types (i.e. email accounts and full accounts)

Figure 56: Definitions

When users perform a lite registration to your site, an email account is created, to which a unique identifier is assigned (UID). When, later down the line, the same email address is used to fully register to your site, CDC automatically assigns it the same UID.

Lite Account

- A new lite account is created every time an non-logged-in user subscribes with an email address

- No other identifiers or social data are associated with this type of account
- Schema-required fields are ignored
- Can contain both profile and custom data fields

Full Account

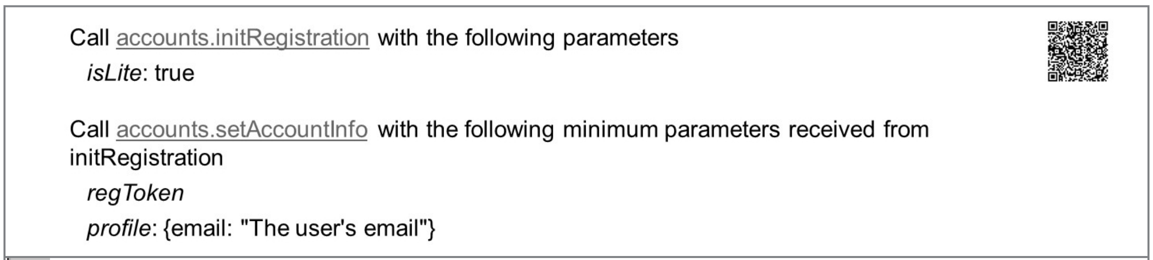
- Created when a site visitor completes the Customer Identity registration flow (regardless of whether or not the account is pending verification)
- These users can register with a social identity or create a password for your site.
- In the context of lite registration, full account data is merged into the corresponding email account every time an accounts.setAccountInfo call is made for that account.

Email Accounts

- Note that data is merged from full accounts or lite registrations into email accounts and not vice versa, with the exception of subscription data that is shared in real-time between account types.
- An email-based view is enabled only when lite registration or subscription management are enabled in your Customer Data Cloud site.

Subscription Object

- Stores subscription data for a designated mailing list
- Note that subscription data is the only type of data shared between all accounts (i.e. in email accounts and full accounts). Therefore, you can retrieve subscription data of fully-registered users and lite registrations.
- Available only with Enterprise Preference Manager


Call accounts.initRegistration with the following parameters

isLite: true

Call accounts.setAccountInfo with the following minimum parameters received from initRegistration

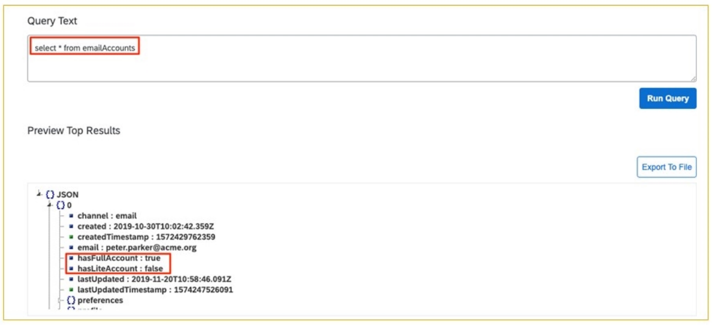
regToken

profile: {email: "The user's email"}

Figure 57: Create Lite Accounts in the API

https://developers.gigya.com/display/GD/Lite+Registration+Quick+Start
+Guide#LiteRegistrationQuickStartGuide-QuickStartAPIBasedLiteRegistration

- A new entity in our system (***emailAccounts***)
- **Merges data** of *full accounts* and *lite accounts* together into a single account based on the *user's email address*
- All accounts have a *UID* associated with them


Query Text

```
select * from emailAccounts
```

```
{} JSON
  {} 0
    channel : email
    created : 2019-10-30T10:02:42.359Z
    createdTimestamp : 1572429762359
    email : peter.parker@acme.org
    hasFullAccount : true
    hasLiteAccount : false
    lastUpdated : 2019-11-20T10:58:46.091Z
    lastUpdatedTimestamp : 1574247526091
    {} preferences
```

Figure 58: Email Account

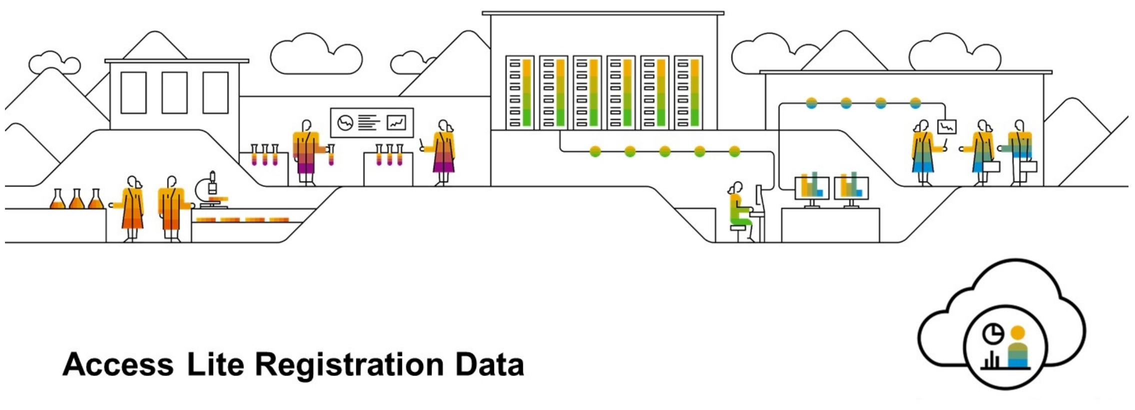


Figure 59: Lite Registration

- You can only use **Identity Query Tool** or **accounts.search** to retrieve lite and subscription data. Find the identity query tool in your admin console under Reports > User Identities.
- Subscription data is available from the *accounts* and *emailAccounts* tables
- From **accounts** you can query only full accounts subscriptions, and from **emailAccounts** you can query both full and lite account subscriptions.
- Query examples
  - Query all users who subscribe to "Sports"
  - Query all subscriptions for AmirBeno@gigya.com
  - Query number of users who subscribed to "Crazy Pet Stories" in April
  - Query all subscribers who don't have a registered account yet

Figure 60: Query Lite and Subscription Data

Watch this video for more details:

https://enable.cx.sap.com/media/1_na0bgda5

You can only use Identity Query Tool or accounts.search to retrieve lite and subscription data. Note that subscription data is available from 'accounts' as well as from 'emailAccounts'.

Query examples:

Retrieve all email accounts:

SELECT * FROM emailAccounts

Retrieve all subscribers to a specific newsletter, from email accounts:

SELECT * FROM emailAccounts WHERE subscriptions.<NEWSLETTER-NAME>.email.isSubscribed=true

Retrieve all subscribers to a specific newsletter, from fully registered accounts:

SELECT * FROM emailAccounts WHERE subscriptions.<NEWSLETTER-NAME>.email.isSubscribed=true AND subscriptions.<NEWSLETTER-NAME>.email.hasFullAccount=true

Retrieve all subscriptions for an email address:

SELECT * FROM emailAccounts WHERE profile.email="<EMAIL-ADDRESS>"

Retrieve all users who were created with Lite Registration and are not yet full users:

SELECT * FROM emailAccounts WHERE hasLiteAccount=true AND hasFullAccount=false

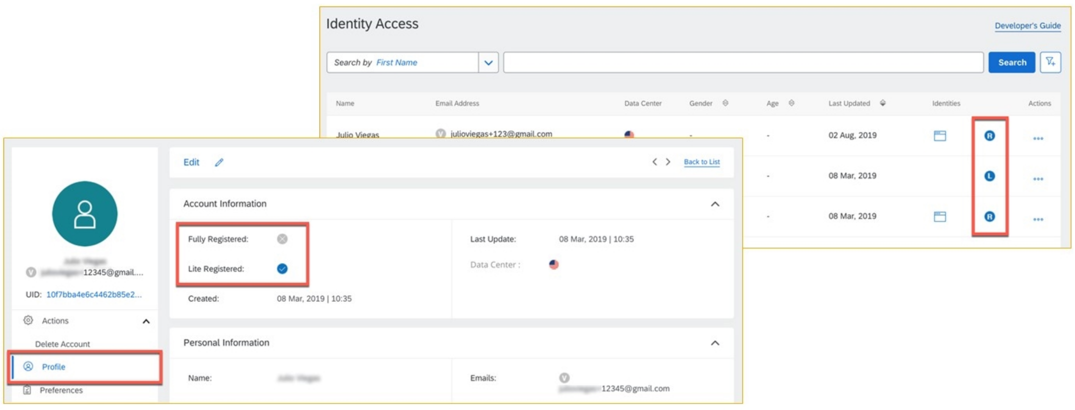
Retrieve from accounts all users who are older than 25

SELECT * FROM accounts WHERE profile.gender="m" AND profile.age > 25

Retrieve all subscribers to a specific newsletter, from email accounts:

SELECT * FROM accounts WHERE subscriptions.<NEWSLETTER-NAME>.email.isSubscribed=true

Identity Access: the dashboard for managing your users in Customer Data Cloud console, displays both lite and fully-registered users.



Name	Email Address	Data Center	Gender	Age	Last Updated	Identities		Actions
Julio Viegas	julioviegas+123@gmail.com		-	-	02 Aug, 2019		R	•••
				-	08 Mar, 2019		L	•••
				-	08 Mar, 2019		R	•••

[redacted]
[redacted]12345@gmail....
UID: 10f7bba4e6c4462b85e2...

Account Information

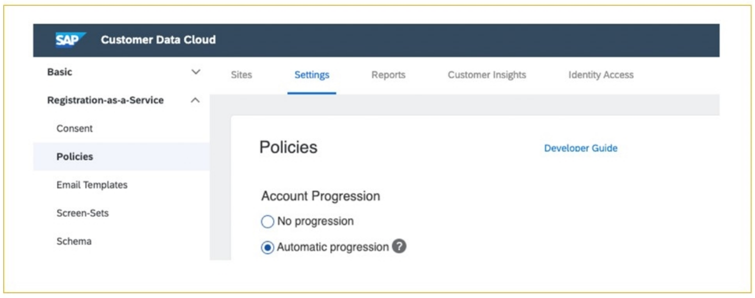
Fully Registered: ✗
Lite Registered: ✓
Created: 08 Mar, 2019 | 10:35
Last Update: 08 Mar, 2019 | 10:35
Data Center :

Personal Information

Name: [redacted]
Emails: [redacted]12345@gmail.com

Figure 61: Identity Access with Email Accounts

- A policy to handle what happens when a lite user with a specific email address performs a full registration on your website with the same email



Policies

Account Progression

No progression [✓ Automatic progression]

- If the policy is set to **auto**, lite data will be copied over to the full account, including consent and subscription information. If the full registration includes data that is different from the existing lite data, the full registration data will be saved to the full account.

Figure 62: Lite Account Progression

Note: Because full registration is a multi-step process, there is some latency between full registration and the data merge.

Exercise 3

Figure 63: Exercise 3

Full Registration

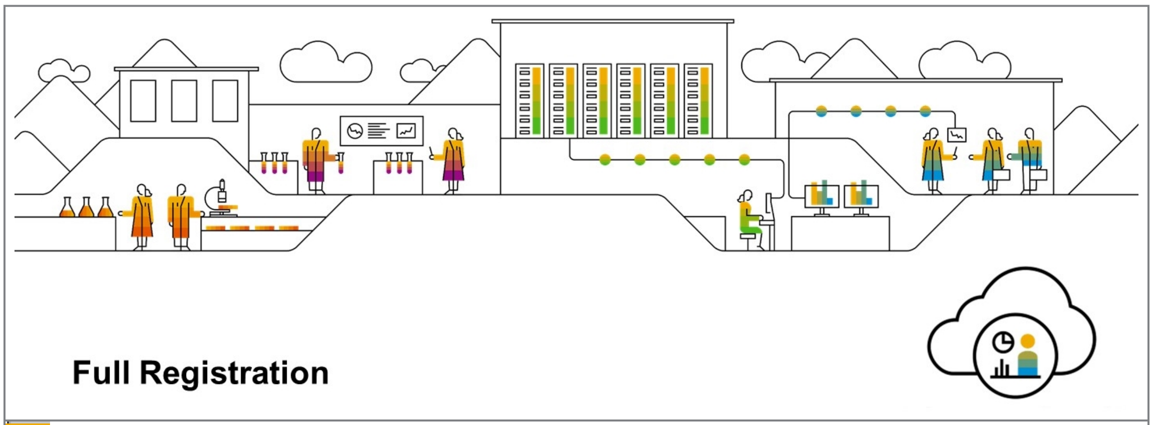
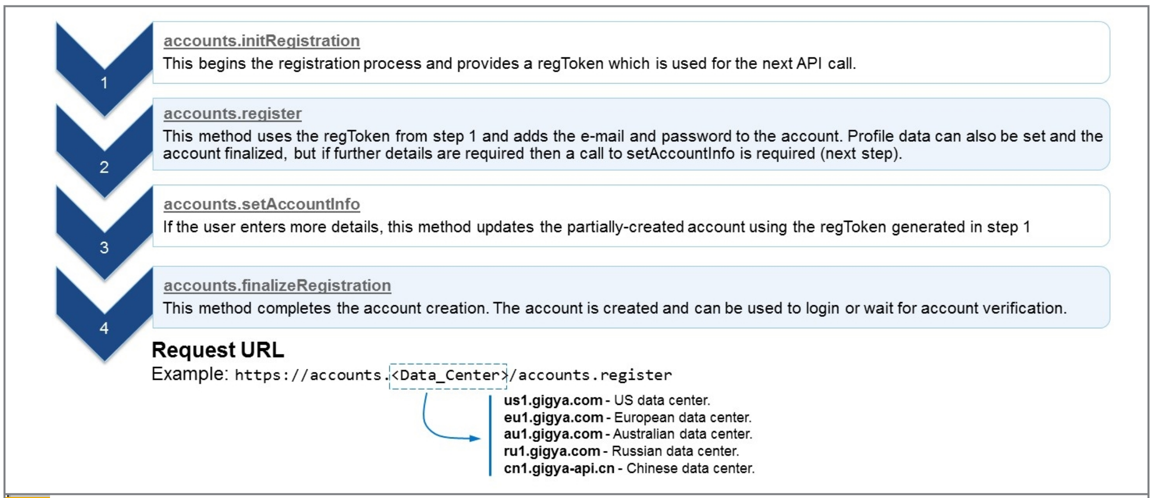


Figure 64: Full Registration



1 **accounts.initRegistration**
This begins the registration process and provides a regToken which is used for the next API call.

2 **accounts.register**
This method uses the regToken from step 1 and adds the e-mail and password to the account. Profile data can also be set and the account finalized, but if further details are required then a call to setAccountInfo is required (next step).

3 **accounts.setAccountInfo**
If the user enters more details, this method updates the partially-created account using the regToken generated in step 1

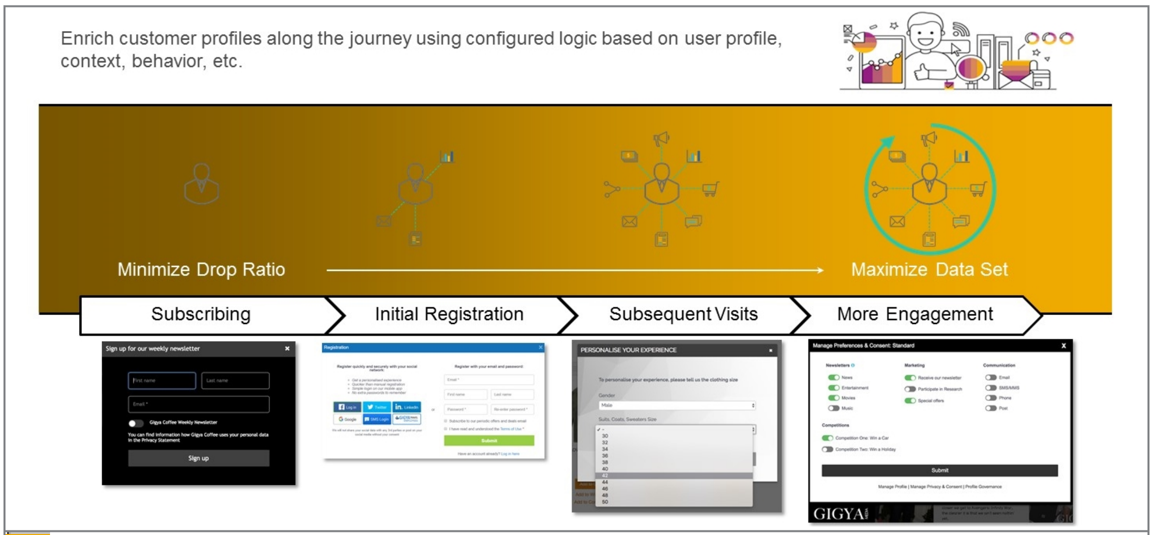
4 **accounts.finalizeRegistration**
This method completes the account creation. The account is created and can be used to login or wait for account verification.

Request URL
Example: `https://accounts.<Data_Center>/accounts.register`

us1.gigya.com - US data center.
eu1.gigya.com - European data center.
au1.gigya.com - Australian data center.
ru1.gigya.com - Russian data center.
cn1.gigya-api.cn - Chinese data center.

Figure 65: Create a Full Account via the API Calls

You cannot use accounts.getAccountInfo to query lite accounts.



Enrich customer profiles along the journey using configured logic based on user profile, context, behavior, etc.

Figure 66: Progressive Profiling

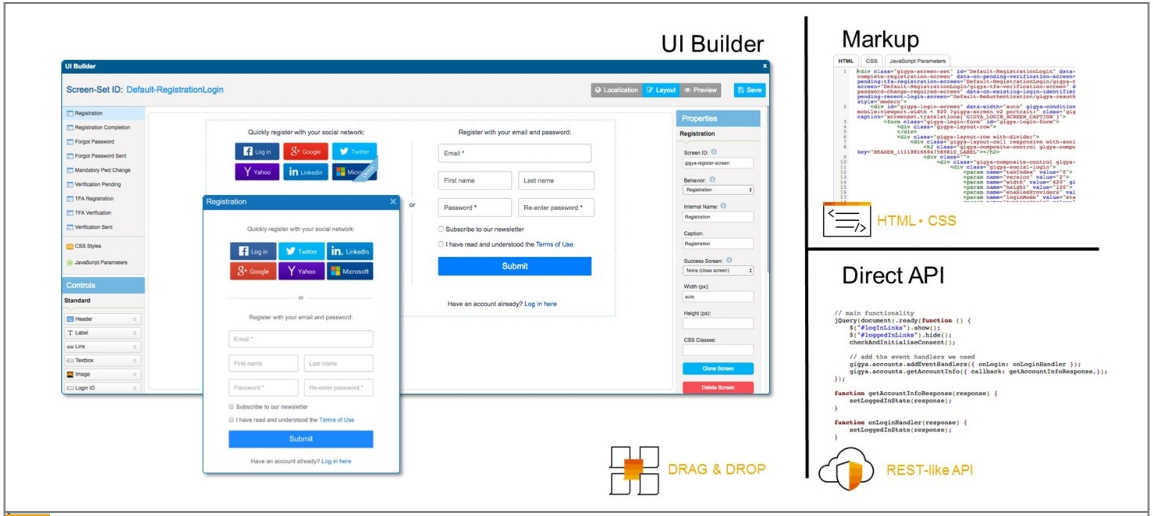


Figure 67: Implementation Approaches

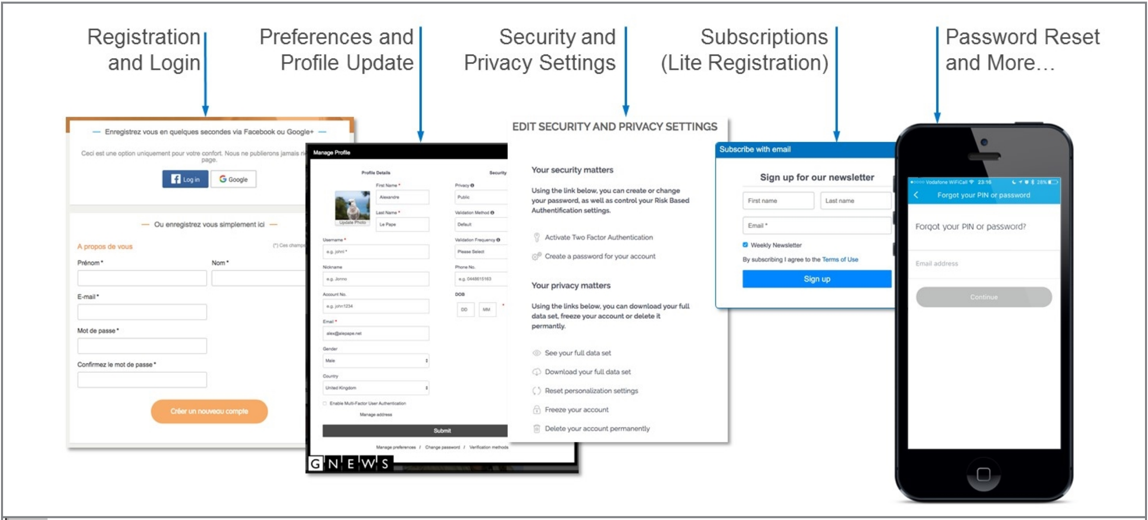

Registration and Login
Preferences and Profile Update
Security and Privacy Settings
Subscriptions (Lite Registration)
Password Reset and More…

Figure 68: Implement All User's Flows Using the UI Builder

Figure 69: Exercise 4

UI Builder

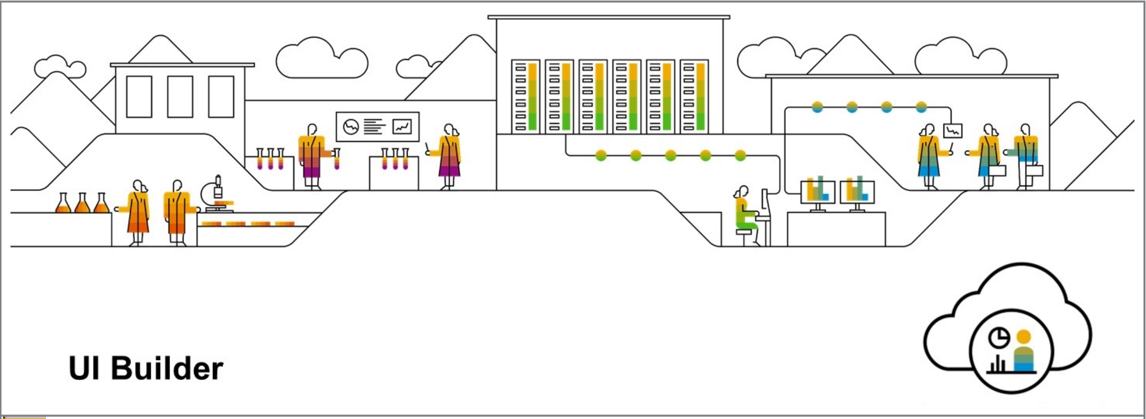


Figure 70: UI Builder

Customer Data Cloud provides registration, login, and other flows out of the box. These flows are called **Screen-Sets**. Each flow contains *multiple screens*.

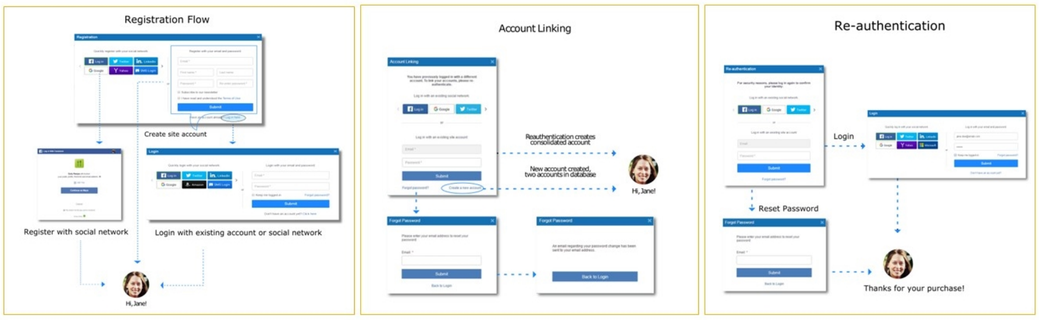

Registration Flow
Create site account
Register with social network
Login with existing account or social network
Hi, Jane!
Account Linking
Reauthentication creates consolidated account
New account created, two accounts in database
Hi, Jane!
Re-authentication
Login
Reset Password
Thanks for your purchase!

Figure 71: Screen-Sets

Update Profile flow is another important Screen-Set

- Initiated by users or when certain conditions are met
- Sub-set of Update Profile flow: **Privacy** and **Communication** screens
- **Change Password** and edit the phone number

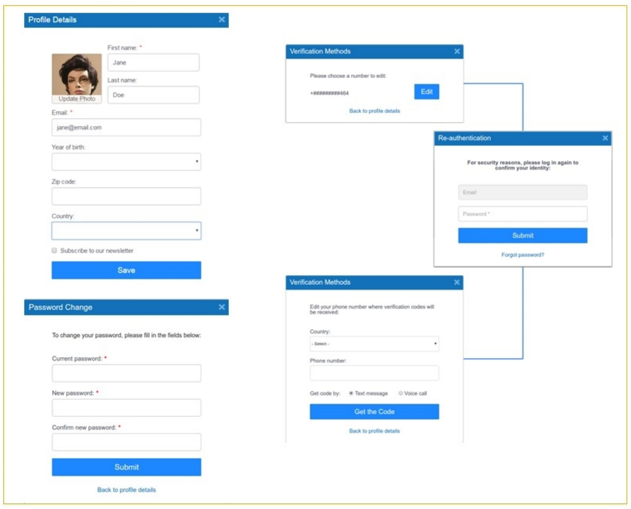
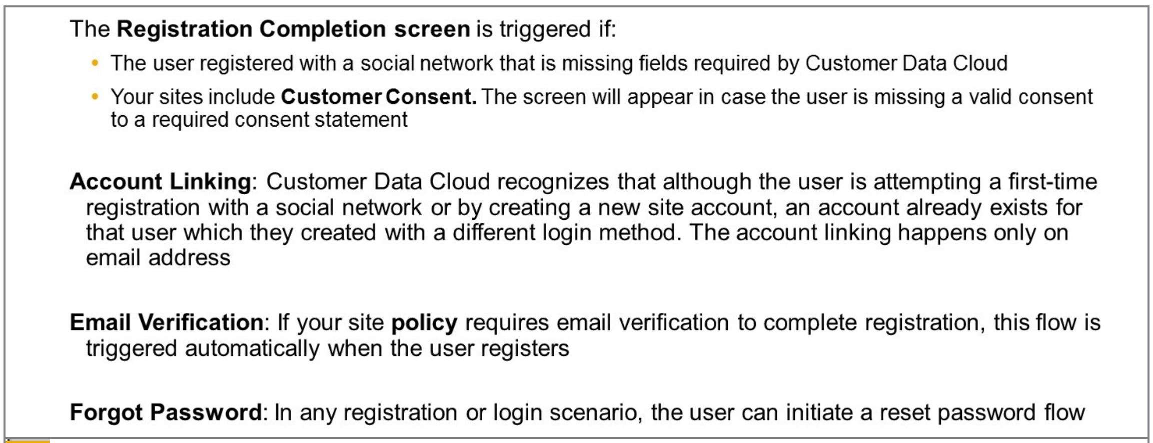


Figure 72: Update Profile Flow

- Initiated by users who wish to display and edit their profile info. Can be called if certain conditions are met, such as in a progressive profiling scenario
- Sub-set of the Update Profile flow screen are the Privacy and Communication screens
- Users are able to change password and, if their site has TFA (two factor authentication) enabled, edit the phone number to which verification codes are sent



The **Registration Completion screen** is triggered if:

- The user registered with a social network that is missing fields required by Customer Data Cloud
- Your sites include **Customer Consent.** The screen will appear in case the user is missing a valid consent to a required consent statement

Account Linking: Customer Data Cloud recognizes that although the user is attempting a first-time registration with a social network or by creating a new site account, an account already exists for that user which they created with a different login method. The account linking happens only on email address

Email Verification: If your site **policy** requires email verification to complete registration, this flow is triggered automatically when the user registers

Forgot Password: In any registration or login scenario, the user can initiate a reset password flow

Figure 73: Advanced Screen Flows

The registration completion screen is triggered if the user registered with a social network that is missing fields which Customer Data Cloud requires. For example, email is a required CDC field, but is not part of the user profile in Twitter, so a prompt asks the user for the minimal required data without impinging the ease of registering with a social network. The user does not yet have a valid session when the screen appears as part of the registration flow.

If your sites includes Customer Consent, the screen will appear if the user is missing a valid consent for a required consent statement. If the user has an active session when this screen appears, the session is terminated. If the user closes the screen without agreeing to the mandatory consent statements (clicks the "X"), the user is logged out (accounts.logout and socialize.logout are fired).

Account Linking: CDC recognizes when the user is attempting a first-time registration with a social network or by creating a new site account but an account already exists for that user which they created with a different login method (e.g., the user is trying to create a new site account but that email is associated with an existing user account created by registering with a social network).

- The Customer Data Cloud UI Builder is a canvas where you edit your Screen-Sets, designed to make implementing Customer Identity as fast and easy as possible.

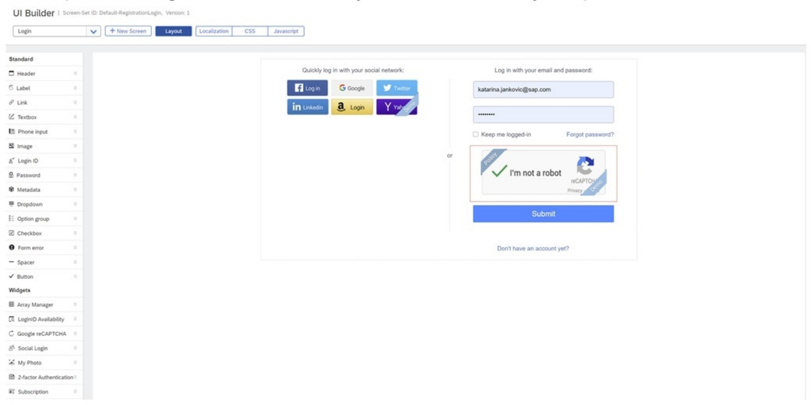
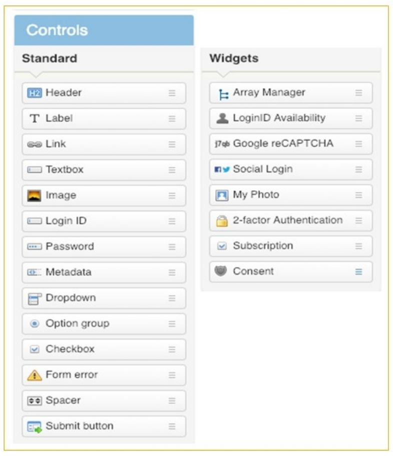


Figure 74: UI Builder

- Easy to use drag-and-drop interface
- Optimized screens for mobile environments
- The UI Builder includes:
- Drag-and-drop visual editing
- Preview on multiple devices
- Localization canvas for handling language translations
- Field mapping that supports a dynamic schema
- CSS and HTML editor for custom designs
- JavaScript editor for handling events related to the screen-set

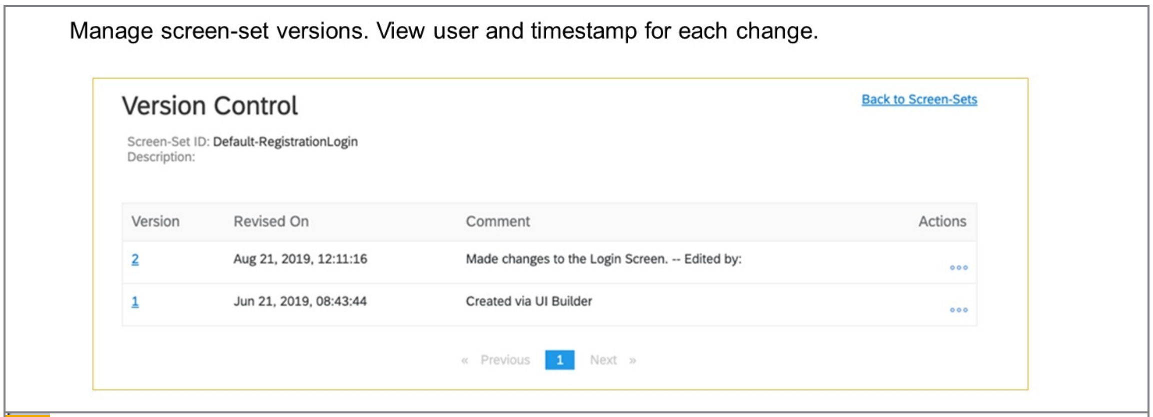


Common controls

- **Add or remove** the controls you require for any screen
- Most Control and Widget **Labels** support basic HTML for styling the text
- You can **Map a field** from the schema
- Set the **Tab index** for all of the controls and widgets
- **Error display name** is used in error messages and it's displayed when the corresponding field is not filled out correctly
- Use the **CSS classes** field to attach a custom CSS class to the field
- **Visible when** property is a JavaScript expression that determines if it is displayed to the user

Figure 75: Standard Controls and Widgets

https://developers.gigya.com/display/GD/UI+Builder#UIBuilder-Controls



Manage screen-set versions. View user and timestamp for each change.

Version Control

Back to Screen-Sets

Screen-Set ID: Default-RegistrationLogin
Description:

Version	Revised On	Comment	Actions
2	Aug 21, 2019, 12:11:16	Made changes to the Login Screen. -- Edited by:	•••
1	Jun 21, 2019, 08:43:44	Created via UI Builder	•••

« Previous 1 Next »

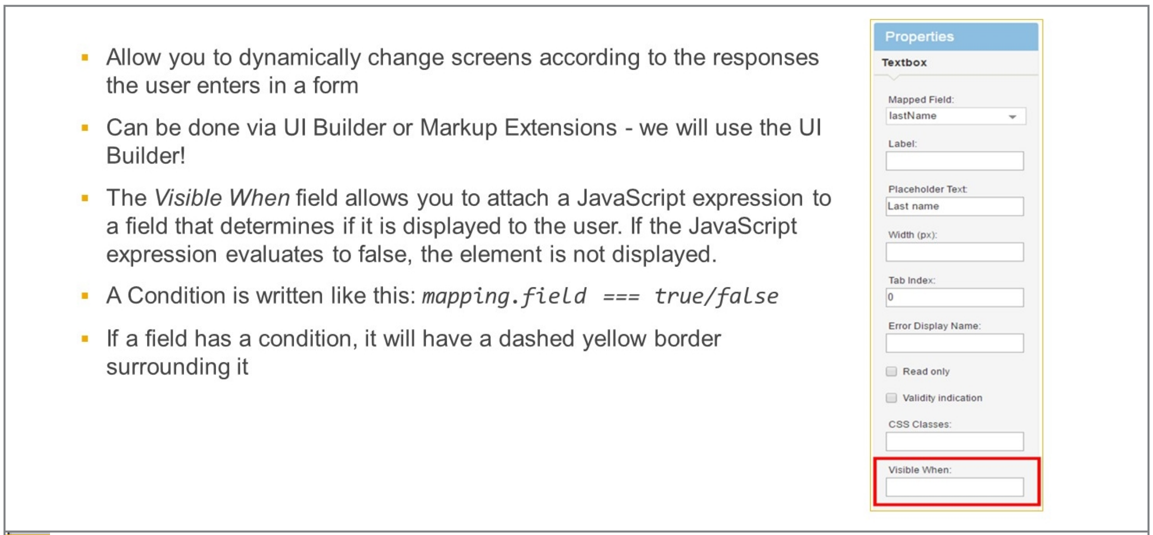
Figure 76: Screen-Set Version Control

Overview:

- Manage screen-set versions. View user and timestamp for each change.
- Provide the option to rollback changes by reverting to older versions and/or opening them for editing.
- Allow clients to audit changes in their registration flow or design.
- Multiple versions can be stored per site.
- Best practice - edit screen-sets on staging site only.

Benefits:

- Update existing screen-sets without having to create new ones.
- Mitigates risk of permanent errors with the ability to undo changes.
- Seasonal or promotional screen-sets can be initiated faster, reverted back to original after expiration, or re-used in the future.
- Version control puts clients at ease - nothing is ever lost.



- Allow you to dynamically change screens according to the responses the user enters in a form
- Can be done via UI Builder or Markup Extensions - we will use the UI Builder!
- The *Visible When* field allows you to attach a JavaScript expression to a field that determines if it is displayed to the user. If the JavaScript expression evaluates to false, the element is not displayed.
- A Condition is written like this: `mapping.field === true/false`
- If a field has a condition, it will have a dashed yellow border surrounding it

Properties

Textbox

Mapped Field: lastName

Label:

Placeholder Text: Last name

Width (px):

Tab Index: 0

Error Display Name:

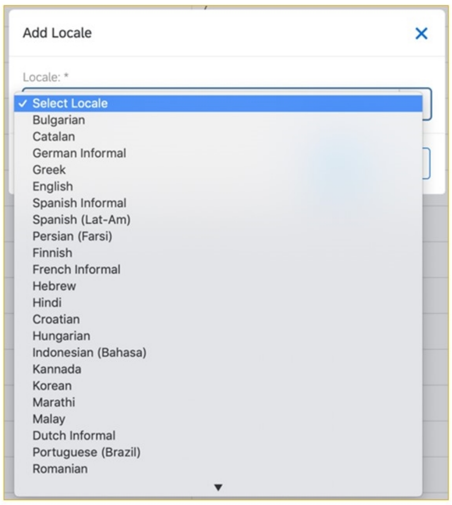
☐ Read only

☐ Validity indication

CSS Classes:

Visible When:

Figure 77: Conditional Workflows



Add Locale

Locale: *

- ✓ Select Locale
- Bulgarian
- Catalan
- German Informal
- Greek
- English
- Spanish Informal
- Spanish (Lat-Am)
- Persian (Farsi)
- Finnish
- French Informal
- Hebrew
- Hindi
- Croatian
- Hungarian
- Indonesian (Bahasa)
- Kannada
- Korean
- Marathi
- Malay
- Dutch Informal
- Portuguese (Brazil)
- Romanian

Steps to Implement:

1. Go to the UI Builder's Localization tab
2. Export as CSV the language you want to translate
3. Add missing translations
4. Save CSV
5. Re-import into console
6. Include the **lang** parameter in the *WebSDK configuration* of your site (change language code accordingly)

```
{
  lang: "da"
}
```

Figure 78: UI Builder Localization

Policies

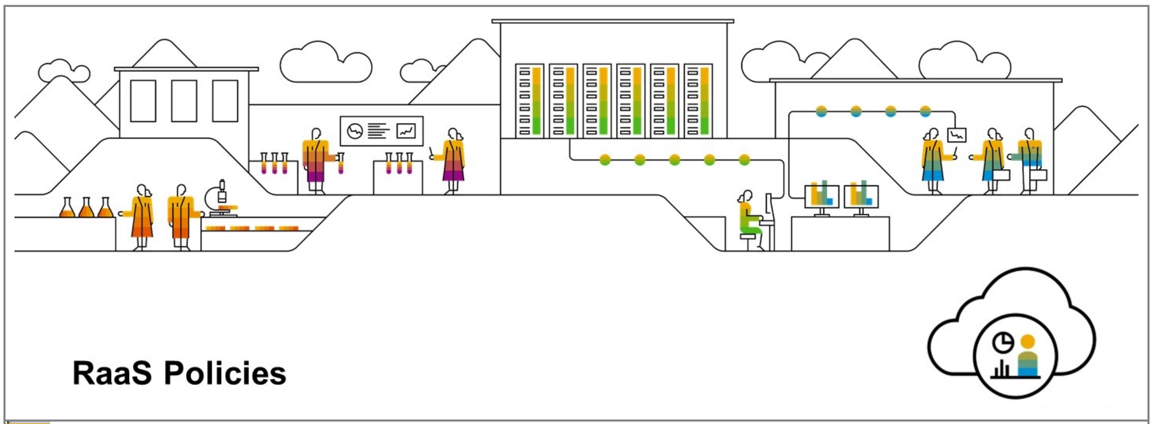
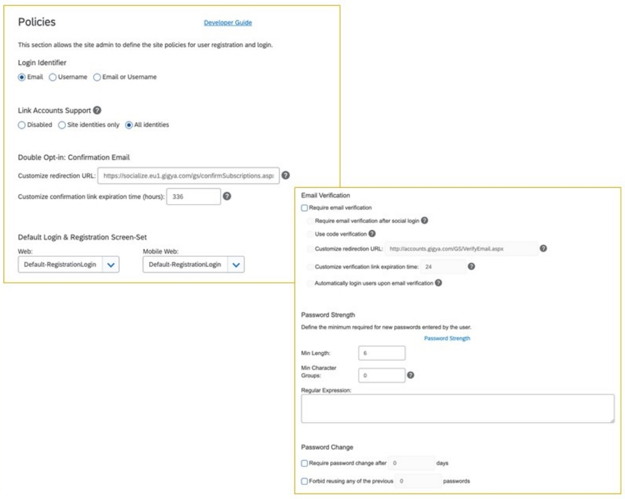


Figure 79: Policies

Simple configuration of key authentication policies

- Customizable password strength
- Primary login identifier
- Automatic account linking
- Email verification
- COPPA age restriction
- Automated emails
- Password change
- Additional security measures



Policies

Developer Guide

This section allows the site admin to define the site policies for user registration and login.

Login Identifier

[✓ Email] Username Email or Username

Link Accounts Support

Disabled Site identities only [✓ All identities]

Double Opt-in: Confirmation Email

Customize redirection URL: https://socialize.eu1.gigya.com/gs/confirmSubscriptions.asp

Customize confirmation link expiration time (hours): 336

Default Login & Registration Screen-Set

Web: Default-RegistrationLogin

Mobile Web: Default-RegistrationLogin

Email Verification

- Require email verification
- Require email verification after social login
- Use code verification
- Customize redirection URL: http://accounts.gigya.com/GS/VerifyEmail.aspx
- Customize verification link expiration time: 24
- Automatically login users upon email verification

Password Strength

Define the minimum required for new passwords entered by the user.

Password Strength

Min Length: 6

Min Character Groups: 0

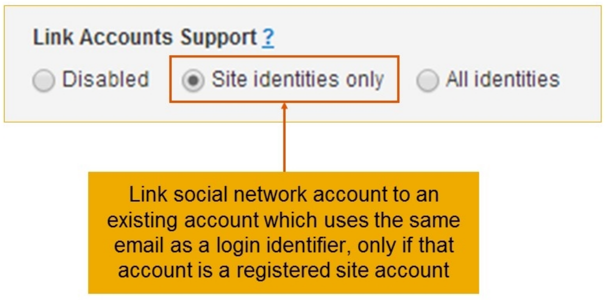
Regular Expression:

Password Change

- Require password change after 0 days
- Forbid reusing any of the previous 0 passwords

Figure 80: Authentication Policies

Account Linking: When two accounts have conflicting login identifiers, CDC assumes that the same user has ownership of both accounts, and enables linking the accounts into a single account, associated with two identities, rather than maintaining two separate accounts

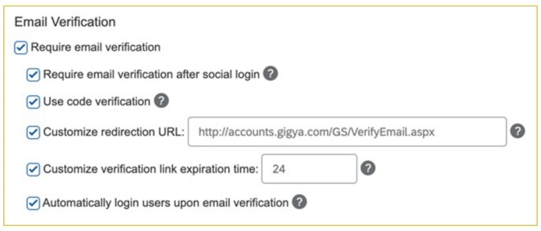

Link Accounts Support ?

Disabled [✓ Site identities only] All identities

Link social network account to an existing account which uses the same email as a login identifier, only if that account is a registered site account

Figure 81: Policies - Link Account Support

- Require account confirmation using a link or code sent over email
- Verification link expiration time: The number of hours that verification emails are valid
- Automatically log in users upon email verification
  - The error code 206005 signals the JavaScript SDK to log the user in using the regToken generated from verifying the email address.
  - Include the Customer Data Cloud JavaScript SDK in the landing page.


Email Verification

☑ Require email verification

☑ Require email verification after social login

☑ Use code verification

☑ Customize redirection URL: http://accounts.gigya.com/GS/VerifyEmail.aspx

☑ Customize verification link expiration time: 24

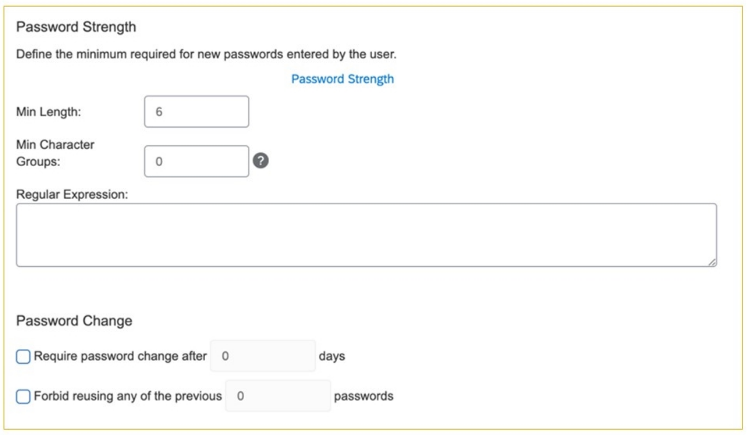
☑ Automatically login users upon email verification

Figure 82: Policies - Email Verification

Error code, 206005, has the message, "Pending Autologin Finalization". This means that when auto-login from the email verification link policy is activated, this response code is passed as the user is redirected to the nextURL specified in the policy. It is not indicative of an error."

Character Groups:

- Capital Letters
- Lowercase letters
- Numbers
- Special Characters


Password Strength

Define the minimum required for new passwords entered by the user.

Password Strength

Min Length:	6
Min Character Groups:	0
Regular Expression:	

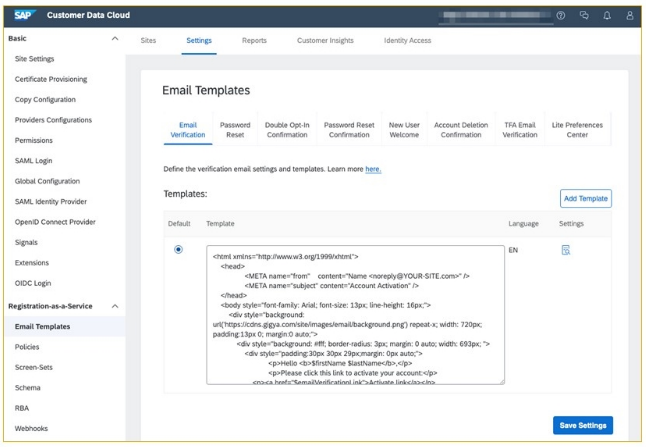
Password Change

☐ Require password change after 0 days

☐ Forbid reusing any of the previous 0 passwords

Figure 83: Policies - Password

- For emails that are sent out to users as part of their site journey
- Fully customizable
- Can be added in multiple languages
- When using site groups, child sites automatically inherit email templates from their parent



Email Templates

Define the verification email settings and templates. Learn more here.

Templates:

Default	Template	Language	Settings
●		EN	

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
         <META name="from"    content="Name <noreply@YOUR-SITE.com>" />
         <META name="subject" content="Account Activation" />
   </head>
   <body style="font-family: Arial; font-size: 13px; line-height: 16px;">
      <div style="background:
url('https://cdns.gigya.com/site/images/email/background.png') repeat-x; width: 720px;
padding:13px 0; margin:0 auto;">
        <div style="background: #fff; border-radius: 3px; margin: 0 auto; width: 693px; ">
          <div style="padding:30px 30px 29px;margin: 0px auto;">
                <p>Hello <b>$firstName $lastName</b>,</p>
                <p>Please click this link to activate your account:</p>
        <p><a href="$emailVerificationLink">Activate link</a></p>
```

Figure 84: Email Templates

Web SDK

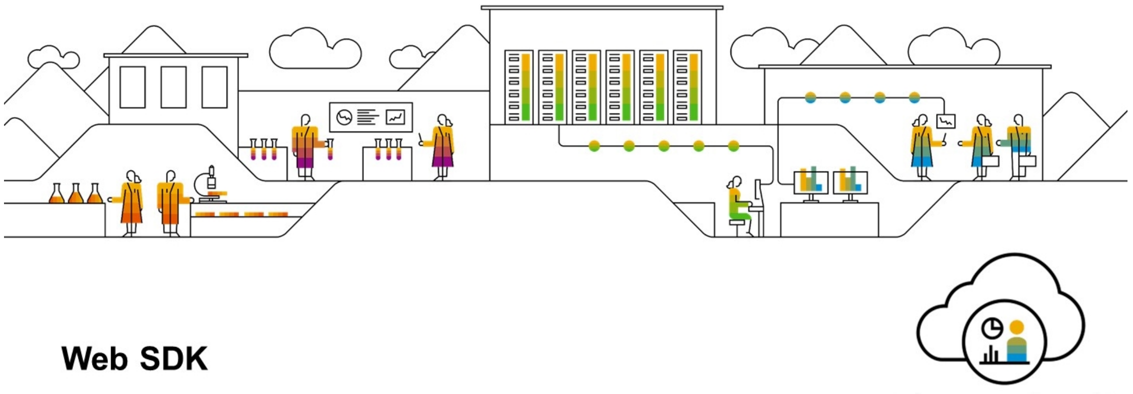
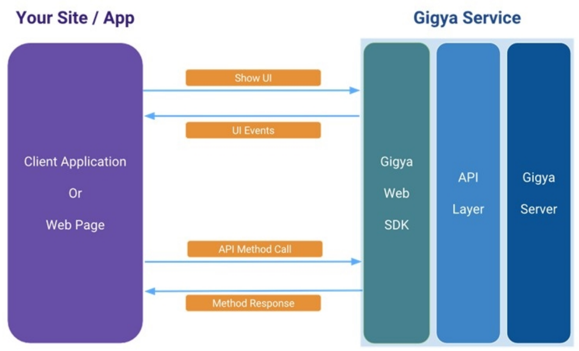


Figure 85: Web SDK



Place the following HTML snippet inside your <head> tag:

<script type="text/javascript" lang="javascript" src="cdns.<DC>/js/gigya.js?apikey=<API_Key>"></script>

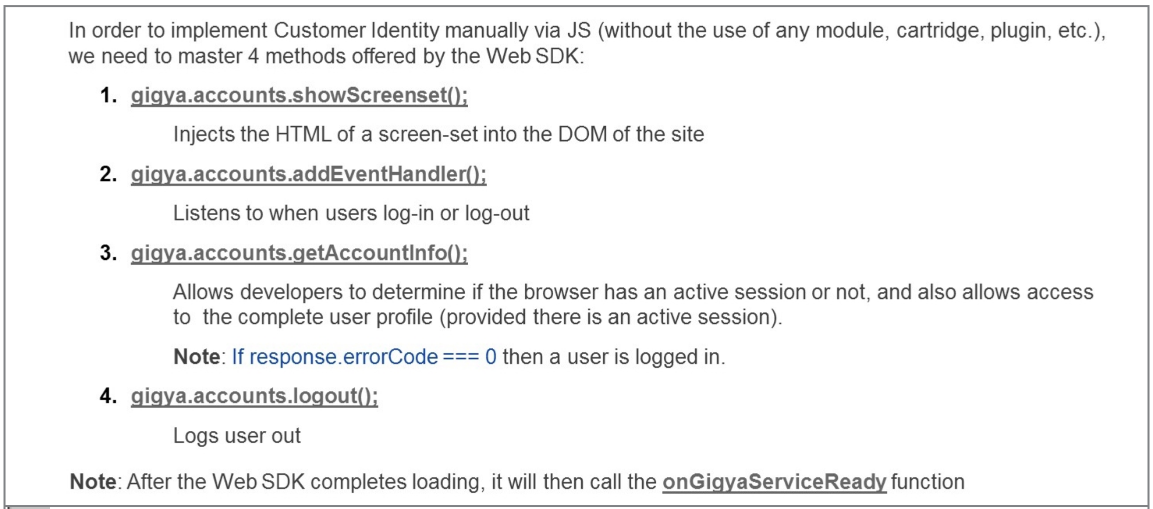
Figure 86: The Web SDK

Where <DC> is one of:

- us1.gigya.com - For the US data center.
- eu1.gigya.com - For the European data center.

- au1.gigya.com - For the Australian data center.
- ru1.gigya.com - For the Russian data center.
- cn1.gigya-api.cn - For the Chinese data center.

By including gigya.js in your code, your javascript code gets access to the gigya.* javascript API namespace.



In order to implement Customer Identity manually via JS (without the use of any module, cartridge, plugin, etc.), we need to master 4 methods offered by the Web SDK:

1. **gigya.accounts.showScreenset();**

   Injects the HTML of a screen-set into the DOM of the site
2. **gigya.accounts.addEventHandler();**

   Listens to when users log-in or log-out
3. **gigya.accounts.getAccountInfo();**

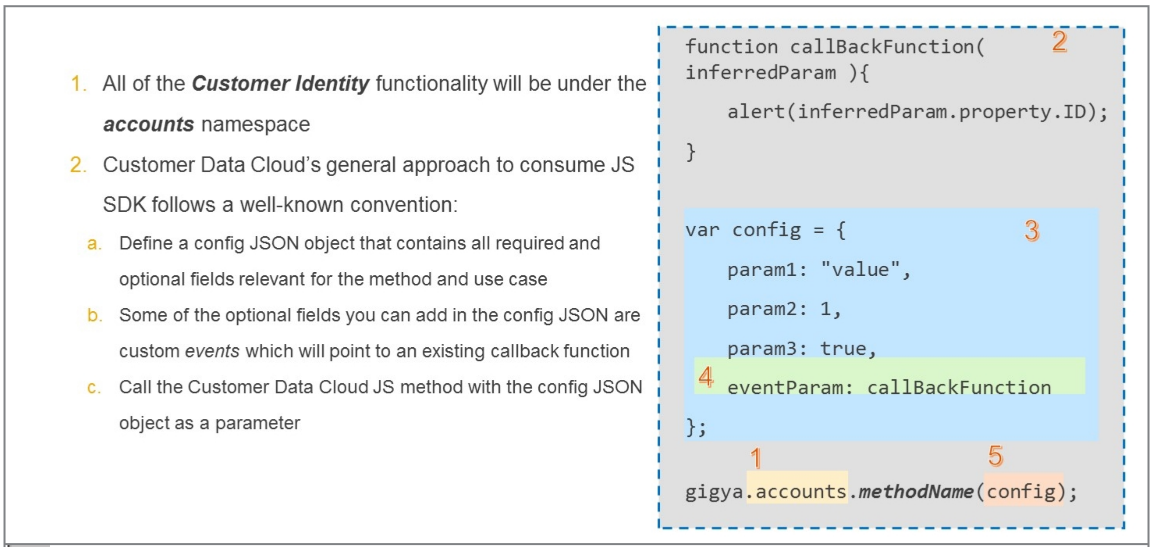
   Allows developers to determine if the browser has an active session or not, and also allows access to the complete user profile (provided there is an active session).

   Note: If response.errorCode === 0 then a user is logged in.
4. **gigya.accounts.logout();**

   Logs user out

Note: After the Web SDK completes loading, it will then call the **onGigyaServiceReady** function

Figure 87: Web SDK Core Methods



1. All of the ***Customer Identity*** functionality will be under the ***accounts*** namespace
2. Customer Data Cloud's general approach to consume JS SDK follows a well-known convention:
   a. Define a config JSON object that contains all required and optional fields relevant for the method and use case
   b. Some of the optional fields you can add in the config JSON are custom *events* which will point to an existing callback function
   c. Call the Customer Data Cloud JS method with the config JSON object as a parameter

```
function callBackFunction(
inferredParam ){
    alert(inferredParam.property.ID);
}

var config = {
    param1: "value",
    param2: 1,
    param3: true,
    eventParam: callBackFunction
};

gigya.accounts.methodName(config);
```

Figure 88: Web SDK Basics

- Customer Data Cloud generates **events** driven by user interactions such as user login, button clicks, etc.
- Applications may register **event handlers** that listen for particular events and execute code when these events are received
- Global application events are generated by the CDC service whenever they occur, regardless of the action that triggered the event
- These are the available global application events:
  - **onLogin** - Fired when a user successfully logs in to Customer Data Cloud
  - **onLogout** - Fired when a user logs out from Customer Data Cloud
  - **onConnectionAdded** - Fired when a user is connected to a provider
  - **onConnectionRemoved** - Fired when a user is disconnected from a provider
  - **onLinkBack** - Fired whenever a linkback is detected

Figure 89: Web SDK Events

https://developers.gigya.com/display/GD/Events#Events-GlobalApplicationevents

https://developers.gigya.com/display/GD/Events#Events-TheonLinkbackEvent

Most common events:

- **onLogin** - Fired when a user successfully logs in to your site.
- **onLogout** - Fired when a user logs out from your site.

Event Handlers are functions following this signature: ***functionName(eventObj)***

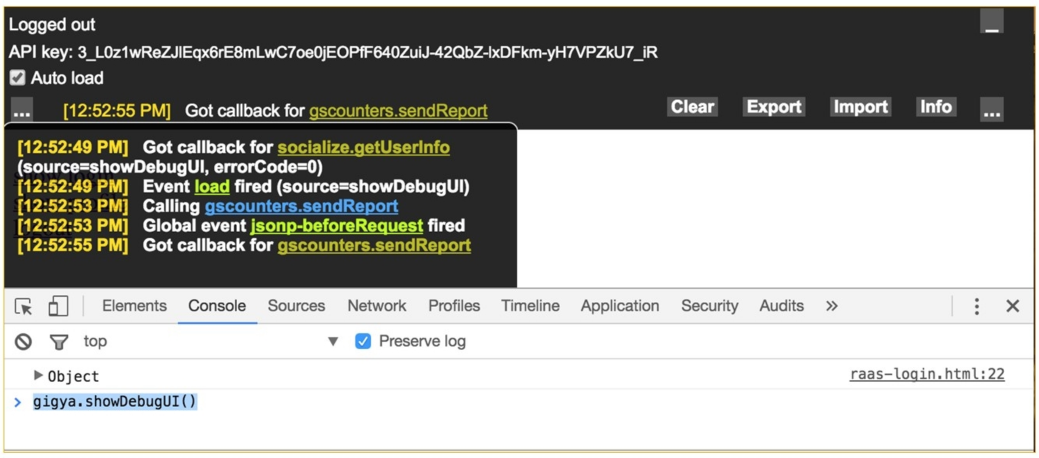
```
function loginEventHandler( eventObj ) {
    loggedIn = true;
    firstName = eventObj.profile.firstName;
    manageUI();
}
function logoutEventHandler( eventObj ) {
    loggedIn = false;
    firstName = "";
    manageUI();
}
```

Registering a Handler for an Event: ***gigya.accounts.addEventHandlers(params)***

```
gigya.accounts.addEventHandlers( {
        onLogin: loginEventHandler,
        onLogout: logoutEventHandler,
} );
```

Figure 90: Adding an event handler function

https://developers.gigya.com/display/GD/Events



```
Logged out
API key: 3_L0z1wReZJlEqx6rE8mLwC7oe0jEOPfF640ZuiJ-42QbZ-lxDFkm-yH7VPZkU7_iR
Auto load
[12:52:55 PM]  Got callback for gscounters.sendReport
[12:52:49 PM]  Got callback for socialize.getUserInfo
(source=showDebugUI, errorCode=0)
[12:52:49 PM]  Event load fired (source=showDebugUI)
[12:52:53 PM]  Calling gscounters.sendReport
[12:52:53 PM]  Global event jsonp-beforeRequest fired
[12:52:55 PM]  Got callback for gscounters.sendReport
```

```
▶ Object                                   raas-login.html:22
> gigya.showDebugUI()
```

Figure 91: Debug with the Web SDK

During development, you may want to know what calls and what responses are taking place in the background.

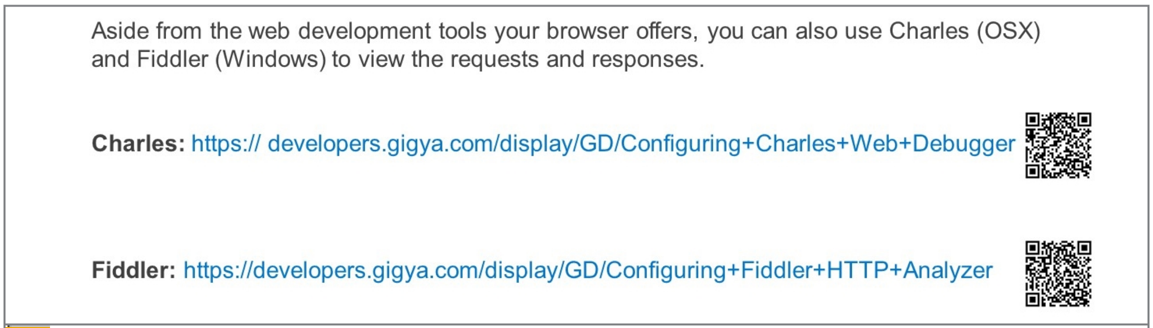
Also, you may encounter an error during the registration process and want to know if Customer Data Cloud provided an error code or a more technical message.

To view this within the browser, Customer Data Cloud has a widget available within the Web SDK:

gigya.showDebugUI();

To use this:

1. Go to your browser's console (on a RaaS-ready page)
   - Chrome: Ctrl + Shift + J (Windows), Cmd + Option + J (Mac)
   - Firefox: Ctrl + Shift + K (Windows), Cmd + Option + K (Mac)
2. Execute the method
3. You will see a widget at the top of the page which will log all calls made to Customer Data Cloud along with the responses



Aside from the web development tools your browser offers, you can also use Charles (OSX) and Fiddler (Windows) to view the requests and responses.

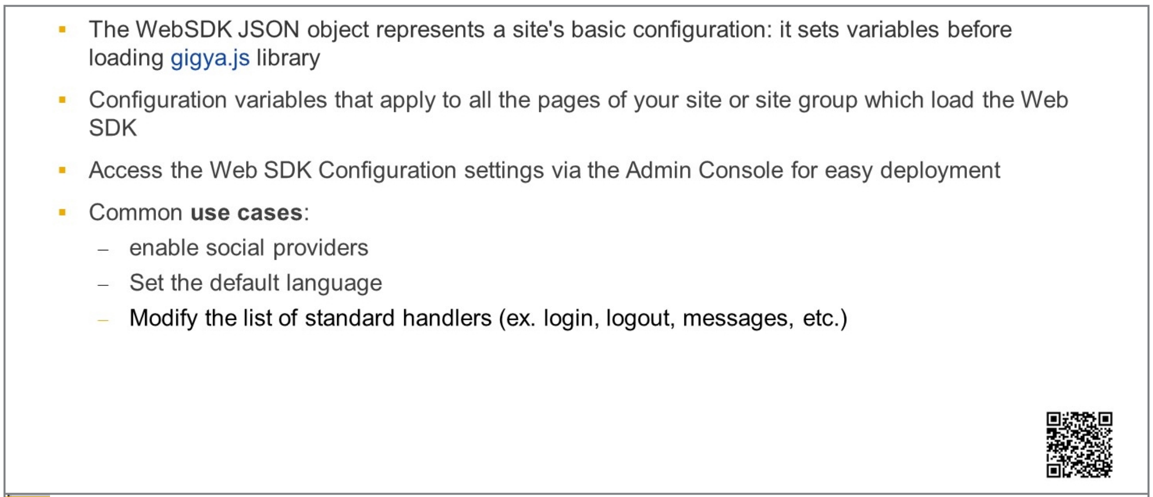
Charles: https:// developers.gigya.com/display/GD/Configuring+Charles+Web+Debugger

Fiddler: https://developers.gigya.com/display/GD/Configuring+Fiddler+HTTP+Analyzer

Figure 92: Alternative Debugging Tools

Charles: https:// developers.gigya.com/display/GD/Configuring+Charles+Web+Debugger

Fiddler: https://developers.gigya.com/display/GD/Configuring+Fiddler+HTTP+Analyzer



- The WebSDK JSON object represents a site's basic configuration: it sets variables before loading gigya.js library
- Configuration variables that apply to all the pages of your site or site group which load the Web SDK
- Access the Web SDK Configuration settings via the Admin Console for easy deployment
- Common **use cases**:
  - enable social providers
  - Set the default language
  - Modify the list of standard handlers (ex. login, logout, messages, etc.)

Figure 93: Web SDK Configuration

Use Cases:

- A standard event map exists by default and is used by gigya.js. It is possible to override it by defining the following method: customEventMap
- Let's define whether to enable or disable the accepted social providers: enabledProviders: '*' where the * means all social providers Example for just Facebook: { enabledProviders: ['facebook'] }
- Force the screen sets to be displayed in a specific language despite a different website default language. An example to force the language of the screen sets in French for France: { lang: "fr-fr" }
- Control the user session expiration. Example { sessionExpiration: -2} (never expires)
- The autoLogin parameter determines if after a user logs in once to the site and visits the same site again, the user is automatically logged in or if the user is logged into Facebook or Google+ at that time
- Create custom buttons for SAML providers
- The user is forced to provide social network credentials during login when forceAuthentication is set to True

Sample code

```
customEventMap: {
eventMap: [{
events: '*', // Allow all events to be called
args: [function(e) {
        // Do whatever you need to do here
         return e;
       }],
```

SessionExpiration values can be:

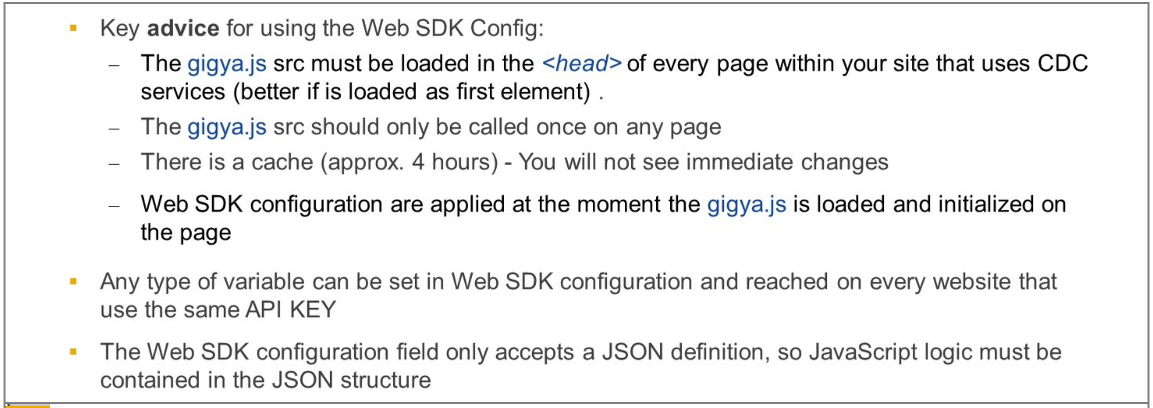
 0: expire when browser closes

-1: Expires after 60 seconds

-2: never expires.

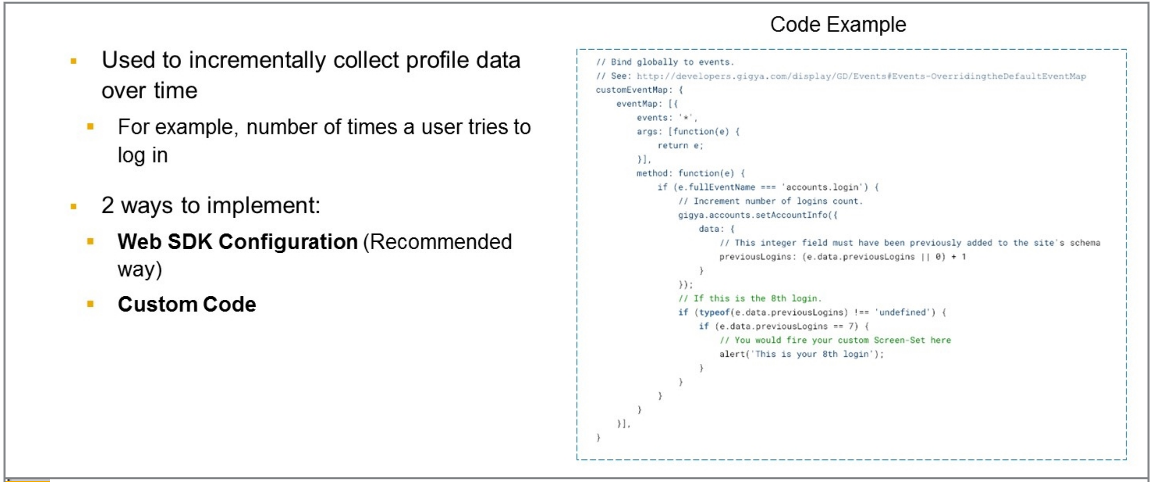
AutoLogin by default is False. It needs to be set on True to enable this feature.

List of all possible attributes: https://developers.gigya.com/display/GD/WebSDK +Configuration



- Key **advice** for using the Web SDK Config:
  - The gigya.js src must be loaded in the *<head>* of every page within your site that uses CDC services (better if is loaded as first element) .
  - The gigya.js src should only be called once on any page
  - There is a cache (approx. 4 hours) - You will not see immediate changes
  - Web SDK configuration are applied at the moment the gigya.js is loaded and initialized on the page
- Any type of variable can be set in Web SDK configuration and reached on every website that use the same API KEY
- The Web SDK configuration field only accepts a JSON definition, so JavaScript logic must be contained in the JSON structure

Figure 94: Web SDK Configuration Tips



- Used to incrementally collect profile data over time
  - For example, number of times a user tries to log in
- 2 ways to implement:
  - **Web SDK Configuration** (Recommended way)
  - **Custom Code**

Code Example

```
// Bind globally to events.
// See: http://developers.gigya.com/display/GD/Events#Events-OverridingtheDefaultEventMap
customEventMap: {
    eventMap: [{
        events: '*',
        args: [function(e) {
            return e;
        }],
        method: function(e) {
            if (e.fullEventName === 'accounts.login') {
                // Increment number of logins count.
                gigya.accounts.setAccountInfo({
                    data: {
                        // This integer field must have been previously added to the site's schema
                        previousLogins: (e.data.previousLogins || 0) + 1
                    }
                });
                // If this is the 8th login.
                if (typeof(e.data.previousLogins) !== 'undefined') {
                    if (e.data.previousLogins == 7) {
                        // You would fire your custom Screen-Set here
                        alert('This is your 8th login');
                    }
                }
            }
        }
    }],
}
```

Figure 95: Progressive Profiling

Web SDK Configuration Example

- Create a global variable in WebSDK Configuration - e.g. loginCount
- Use the custom event map to add a function in the onLogin global event, to increment the count
- Map the loginCount to a data field - using the metadata property on the screen-sets
- In UI builder, use the visible when property to show the relevant fields after the 2nd login or when loginCount == 2

LESSON SUMMARY

You should now be able to:

- Describe the SAP Customer Identity
- Understand Lite Registration and Full Registration

UNIT 5 REST API

UNIT OBJECTIVES

- Use the REST API

Unit 5
Lesson 1

Understanding the REST API

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use the REST API

REST API

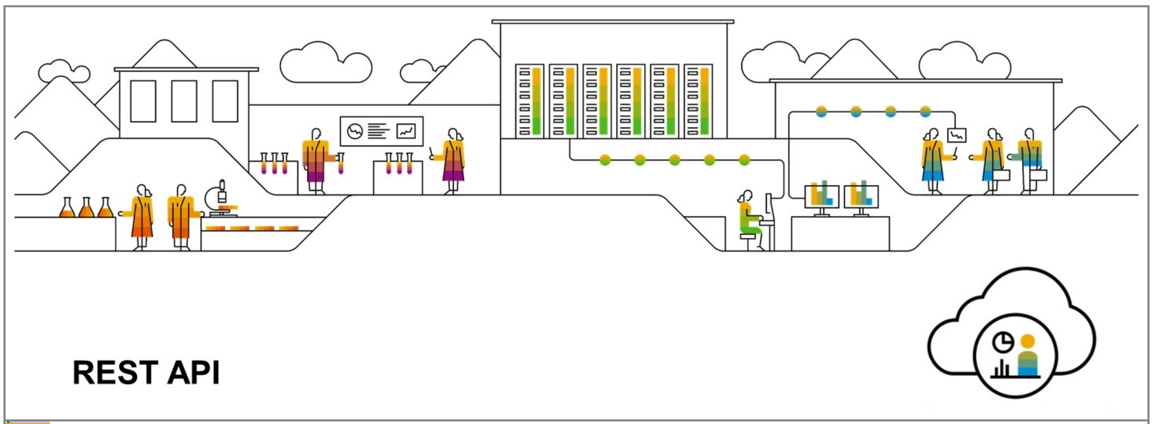
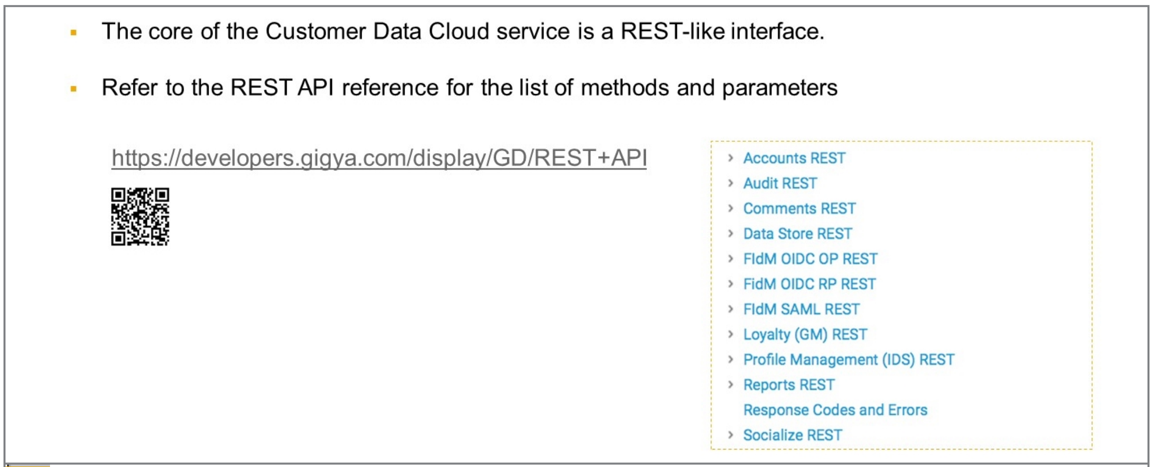


Figure 96: REST API



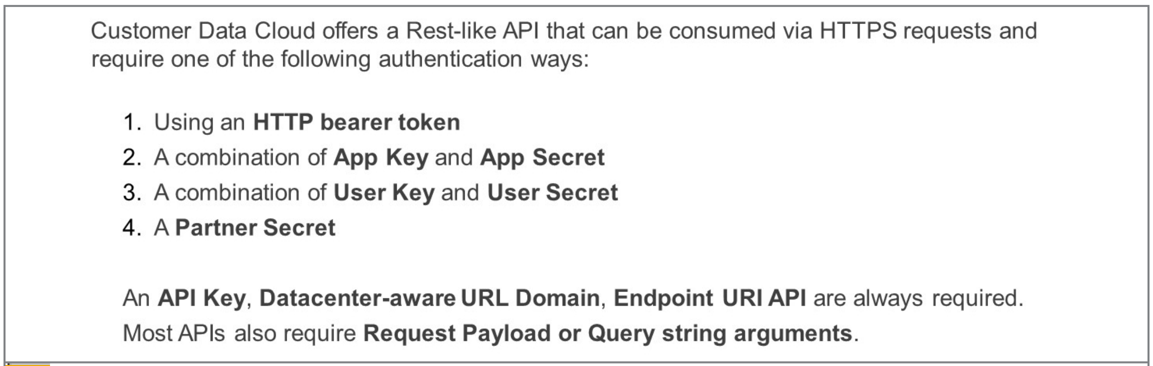
- The core of the Customer Data Cloud service is a REST-like interface.
- Refer to the REST API reference for the list of methods and parameters

https://developers.gigya.com/display/GD/REST+API

- Accounts REST
- Audit REST
- Comments REST
- Data Store REST
- FIdM OIDC OP REST
- FidM OIDC RP REST
- FIdM SAML REST
- Loyalty (GM) REST
- Profile Management (IDS) REST
- Reports REST
- Response Codes and Errors
- Socialize REST

Figure 97: REST API

https://developers.gigya.com/display/GD/REST+API



Customer Data Cloud offers a Rest-like API that can be consumed via HTTPS requests and require one of the following authentication ways:

1. Using an **HTTP bearer token**
2. A combination of **App Key** and **App Secret**
3. A combination of **User Key** and **User Secret**
4. A **Partner Secret**

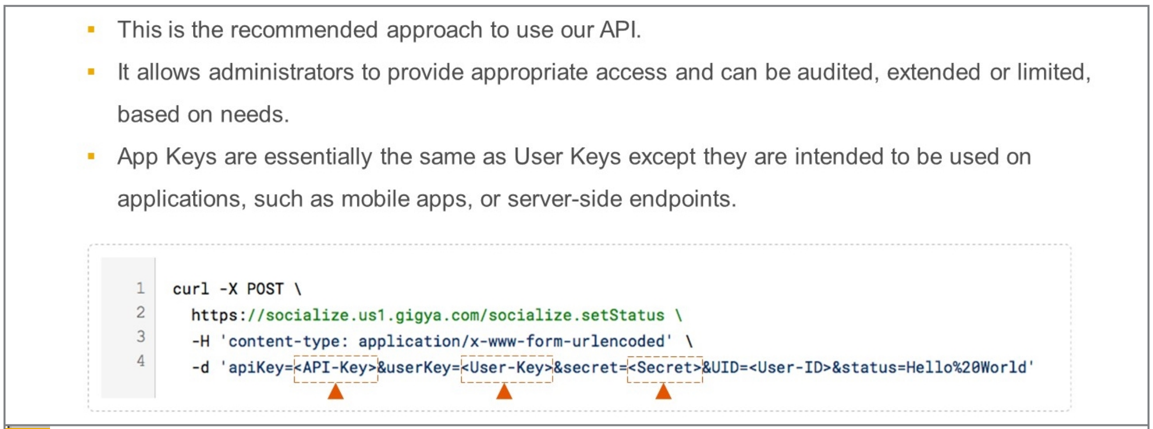
An **API Key**, **Datacenter-aware URL Domain**, **Endpoint URI API** are always required. Most APIs also require **Request Payload or Query string arguments**.

Figure 98: How to Make API Calls

Regarding authentication ways, the above list is ordered accordingly with best practices, with item 1 being the best way and 4 the worse (in fact you must avoid it).

Check this link for more details

https://developers.gigya.com/display/GD/REST+APIs+with+the+Gigya+Authorization +Method



- This is the recommended approach to use our API.
- It allows administrators to provide appropriate access and can be audited, extended or limited, based on needs.
- App Keys are essentially the same as User Keys except they are intended to be used on applications, such as mobile apps, or server-side endpoints.

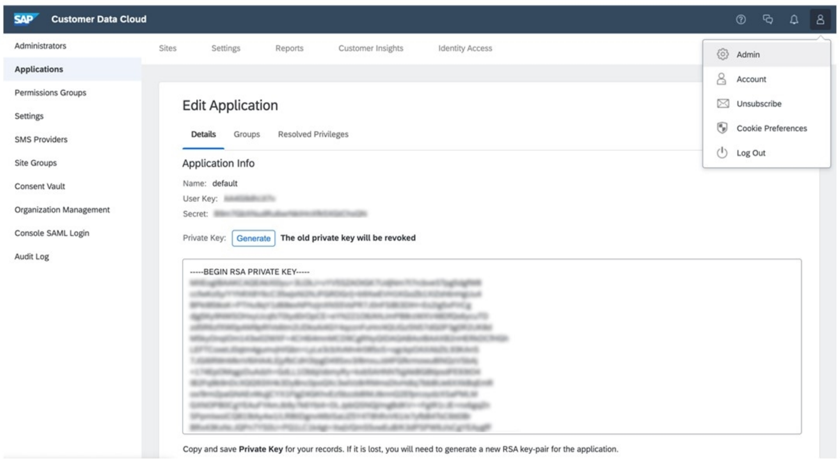
```
curl -X POST \
  https://socialize.us1.gigya.com/socialize.setStatus \
  -H 'content-type: application/x-www-form-urlencoded' \
  -d 'apiKey=<API-Key>&userKey=<User-Key>&secret=<Secret>&UID=<User-ID>&status=Hello%20World'
```

Figure 99: API Key, App/User Key, App/User Secret Key

You can create as many User Keys as needed by inviting users or by creating App Keys (the better way) on the Admin section.

App Key or User Keys have higher API rate limits and given enough notice, can be set to even higher limits if traffic spike are expected.

Using Authorization: Bearer HTTP header instead of Key/Secret pair



Name: default
User Key: [redacted]
Secret: [redacted]
Private Key: Generate The old private key will be revoked

-----BEGIN RSA PRIVATE KEY-----
[redacted]

Copy and save **Private Key** for your records. If it is lost, you will need to generate a new RSA key-pair for the application.

Figure 100: Bearer HTTP token

From https://developers.gigya.com/display/GD/Signing+Requests+to+SAP+Customer+Data+Cloud#SigningRequeststoSAPCustomerDataCloud-AsymmetricKeys

"When creating a new application in the Console, an asymmetric RSA key-pair is assigned to the application. Copy the private key and store it securely."

You can sign an API request to SAP Customer Data Cloud using an HTTP bearer token. This replaces the application/user key and secret signature method. To do so, you should create a JWT, then hash the JWT using the RSA key to create a bearer token and sign the API request using that token.

e.g. Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ilt1c2Vya2V5XSJ9.eyJpYXQiOjE1NDgxOsdkj(the token above is the JWT token itself)

The link on the top contains instructions on how to use the RSA private key to generate the bearer token.

Cheat tool to make calls: https://tools.gigya-cs.com/api/

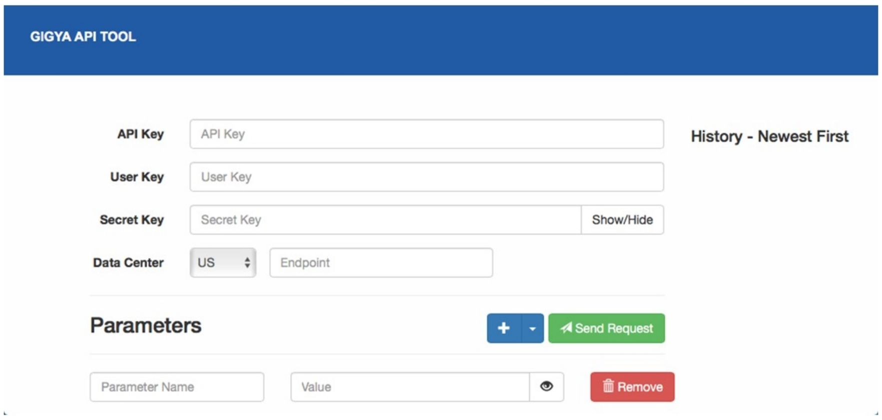
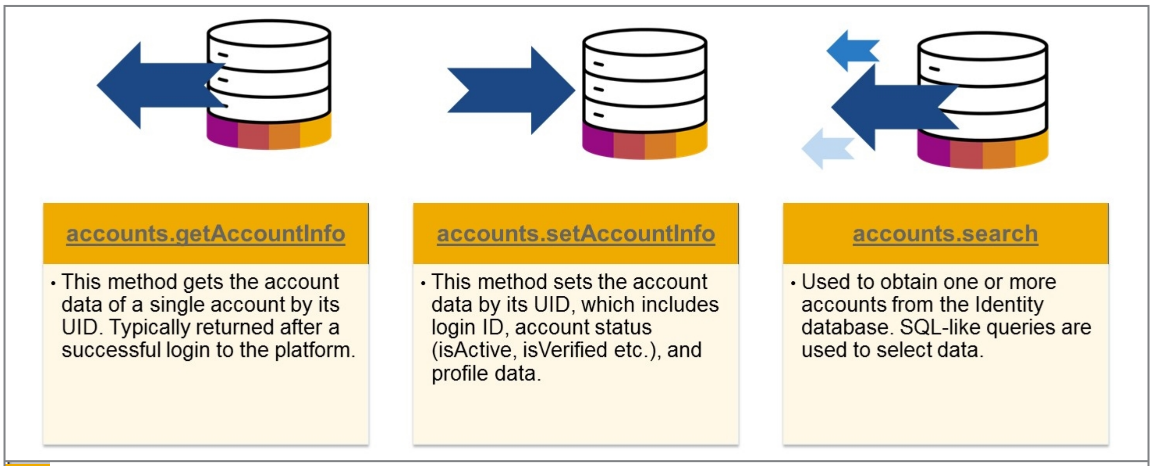


Figure 101: API TOOL

https://tools.gigya-cs.com/api/



accounts.getAccountInfo

- This method gets the account data of a single account by its UID. Typically returned after a successful login to the platform.

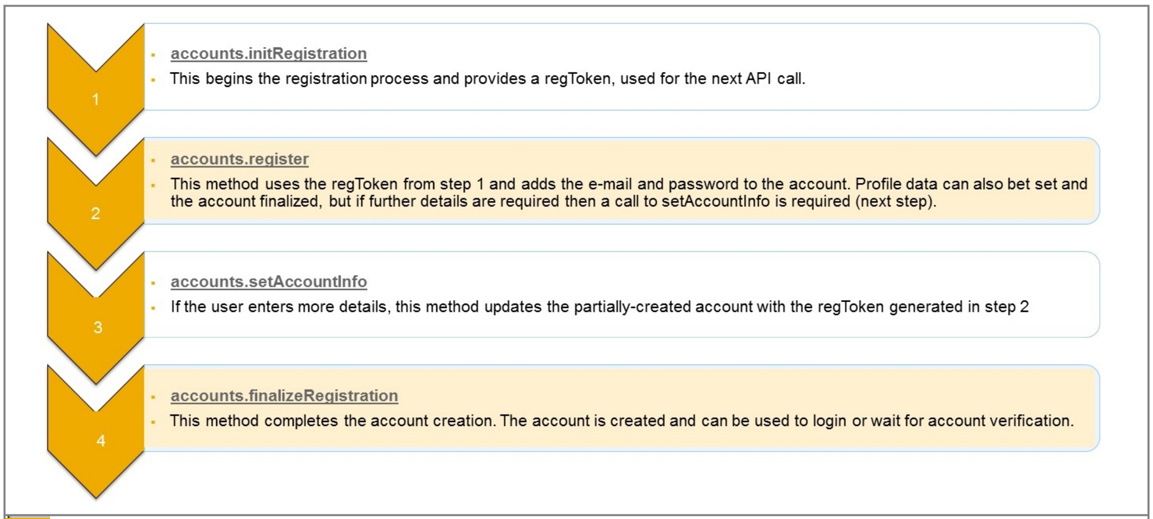
accounts.setAccountInfo

- This method sets the account data by its UID, which includes login ID, account status (isActive, isVerified etc.), and profile data.

accounts.search

- Used to obtain one or more accounts from the Identity database. SQL-like queries are used to select data.

Figure 102: Setting and Getting Account Data



1

- accounts.initRegistration
- This begins the registration process and provides a regToken, used for the next API call.

2

- accounts.register
- This method uses the regToken from step 1 and adds the e-mail and password to the account. Profile data can also bet set and the account finalized, but if further details are required then a call to setAccountInfo is required (next step).

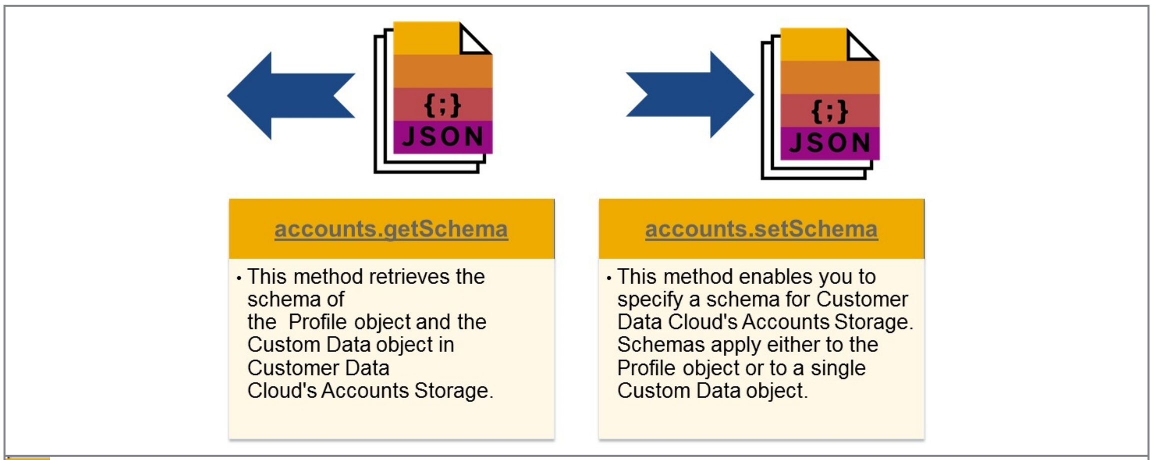
3

- accounts.setAccountInfo
- If the user enters more details, this method updates the partially-created account with the regToken generated in step 2

4

- accounts.finalizeRegistration
- This method completes the account creation. The account is created and can be used to login or wait for account verification.

Figure 103: Core API Flows - Site Registration



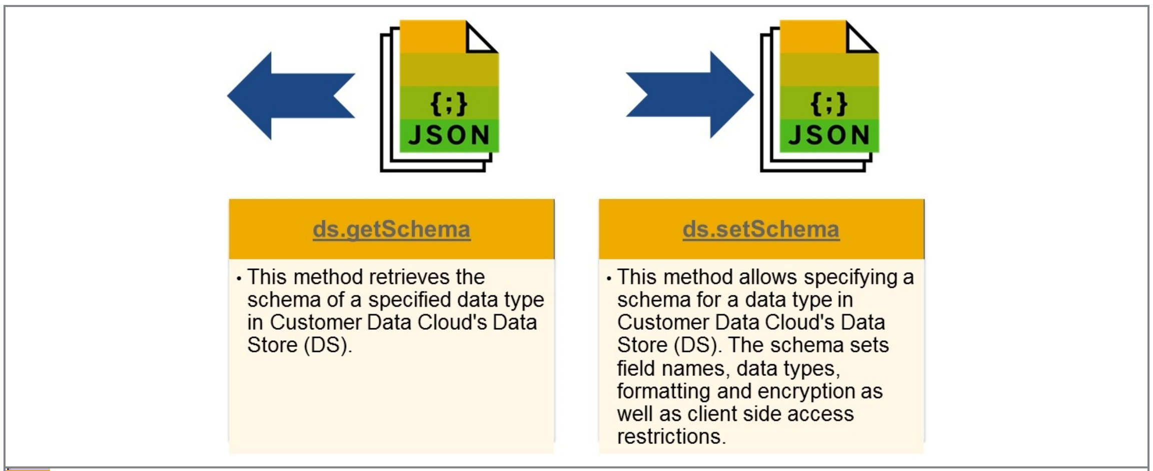
accounts.getSchema

- This method retrieves the schema of the Profile object and the Custom Data object in Customer Data Cloud's Accounts Storage.

accounts.setSchema

- This method enables you to specify a schema for Customer Data Cloud's Accounts Storage. Schemas apply either to the Profile object or to a single Custom Data object.

Figure 104: Reading and Changing Schema for Account Store



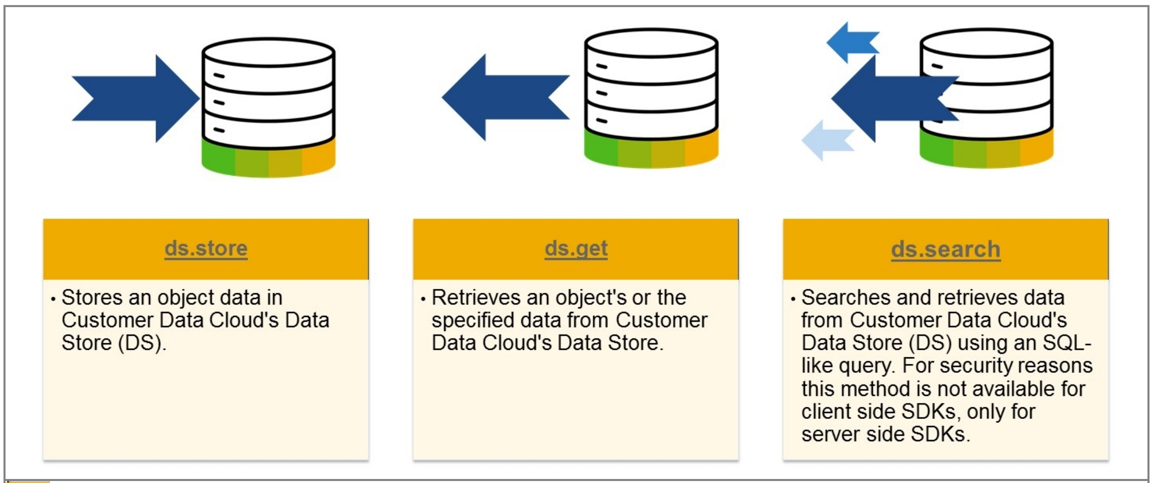
ds.getSchema

- This method retrieves the schema of a specified data type in Customer Data Cloud's Data Store (DS).

ds.setSchema

- This method allows specifying a schema for a data type in Customer Data Cloud's Data Store (DS). The schema sets field names, data types, formatting and encryption as well as client side access restrictions.

Figure 105: Reading and Changing Schema for (Object) Data Store



ds.store

- Stores an object data in Customer Data Cloud's Data Store (DS).

ds.get

- Retrieves an object's or the specified data from Customer Data Cloud's Data Store.

ds.search

- Searches and retrieves data from Customer Data Cloud's Data Store (DS) using an SQL-like query. For security reasons this method is not available for client side SDKs, only for server side SDKs.

Figure 106: Setting and Getting Object Data

Figure 107: Exercise 5

LESSON SUMMARY

You should now be able to:

- Use the REST API

UNIT 6 Authentication Methods

UNIT OBJECTIVES

- Explain the various authentication methods available for the SAP Customer Data Cloud developer

Unit 6
Lesson 1

Understanding Authentication Methods

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the various authentication methods available for the SAP Customer Data Cloud developer

Social Login

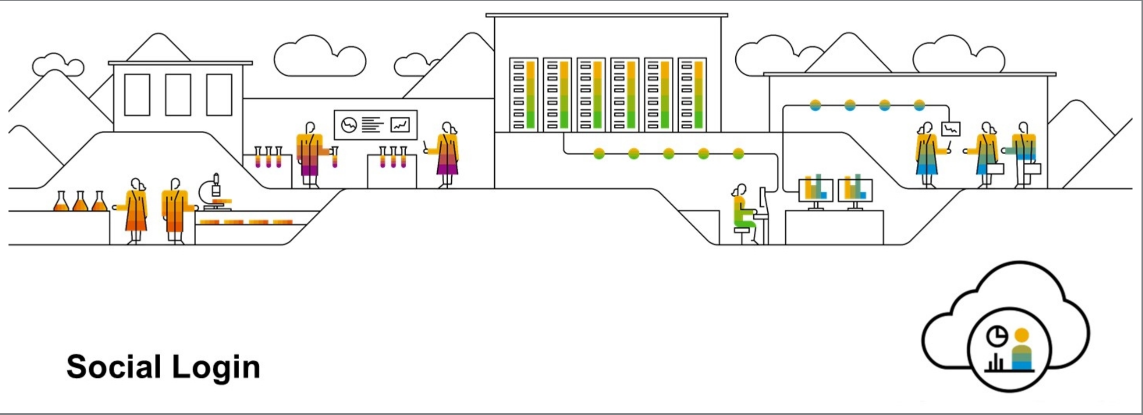
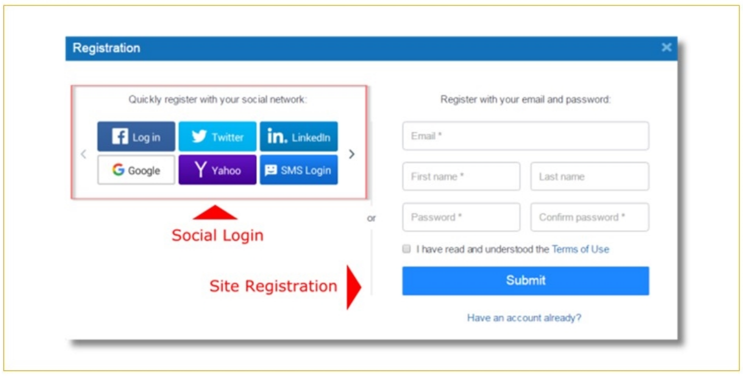


Figure 108: Social Login

Customer Data Cloud's **Social Login** is an authentication system that allows users to register and login to your site using their social network accounts such as Facebook, Twitter, Google, Yahoo, LinkedIn, and others.



Registration

Quickly register with your social network:

Log in | Twitter | LinkedIn | Google | Yahoo | SMS Login

or

Register with your email and password:

Email *

First name * | Last name

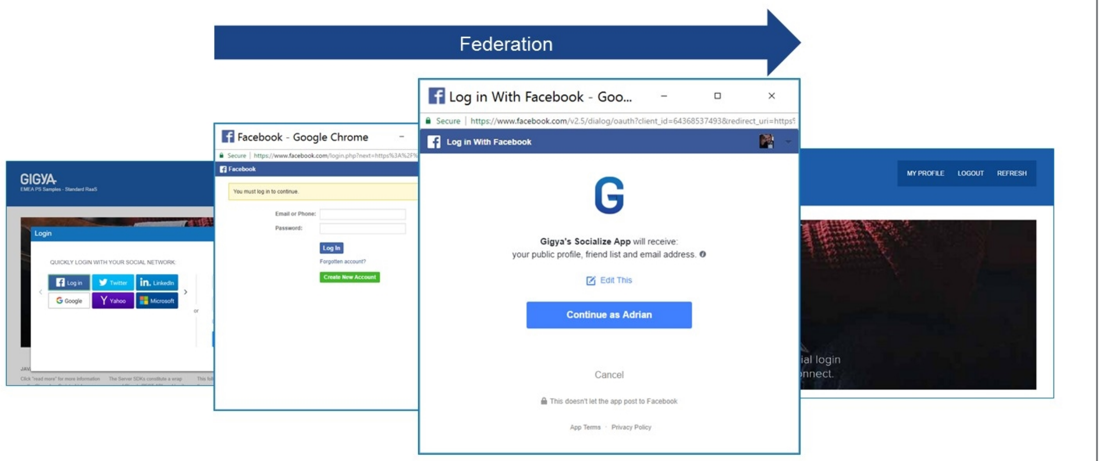
Password * | Confirm password *

I have read and understood the Terms of Use

Submit

Have an account already?

Figure 109: Social Login



Gigya's Socialize App will receive: your public profile, friend list and email address.

This doesn't let the app post to Facebook

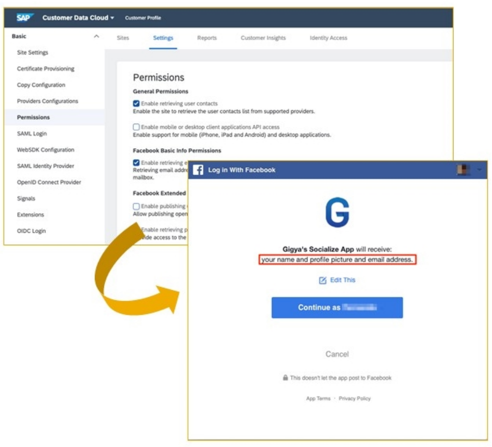
Figure 110: Social Login Example

- The Socialize namespace of functions in the Web SDK allows:
  - connect the user to a social network
  - notify the CDC service that the user has been logged in by the site
  - retrieve extended information regarding the current user
  - authenticate the user using an external provider
  - logs out the current user of the CDC platform.
- Social users *do not have a password*
- If users have a social account but want to login with email + password, the **password reset API** needs to be called
- This will create a **site** identity the user can use to login

Figure 111: Social Users

https://developers.gigya.com/display/GD/Socialize+JS

- Customer Data Cloud Permissions allows you to manage the permissions you have for different social networks, including general and Facebook-related permissions
- There are two type of permissions:
  - *General Permissions*
  - *Specific to a Social Network Permissions*



Enable retrieving user contacts
Enable the site to retrieve the user contacts list from supported providers.

Enable mobile or desktop client applications API access
Enable support for mobile (iPhone, iPad and Android) and desktop applications.

Gigya's Socialize App will receive: your name and profile picture and email address.

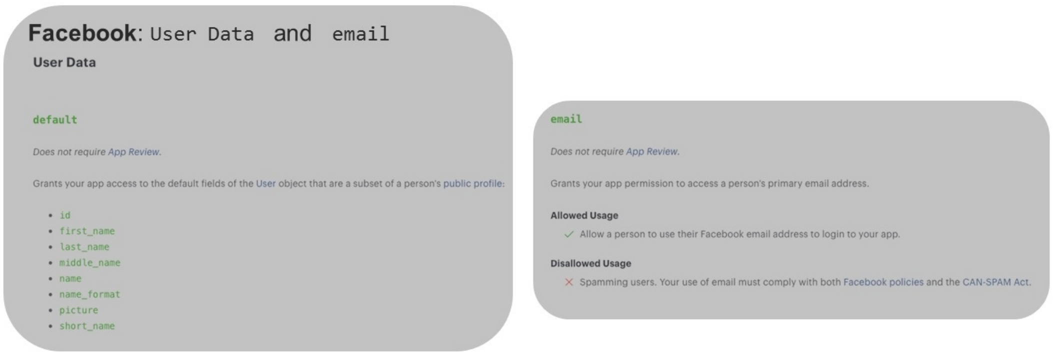
This doesn't let the app post to Facebook

Figure 112: Permissions

- Facebook, Instagram, and LinkedIn require approval for extended data permissions
- To be approved to retrieve user data of extended permissions, you will need to go through a submission process with each of these social providers

- Social providers will require you to prove a use case for your website or application for each extended permission requested and for you to show how it benefits the end user's experience on your site or application

https://developers.facebook.com/docs/facebook-login/permissions#reference-default



Facebook: `User Data` and `email`

User Data

`default`

Does not require App Review.

Grants your app access to the default fields of the User object that are a subset of a person's public profile:

- `id`
- `first_name`
- `last_name`
- `middle_name`
- `name`
- `name_format`
- `picture`
- `short_name`

`email`

Does not require App Review.

Grants your app permission to access a person's primary email address.

Allowed Usage

✓ Allow a person to use their Facebook email address to login to your app.

Disallowed Usage

✗ Spamming users. Your use of email must comply with both Facebook policies and the CAN-SPAM Act.

https://developers.facebook.com/docs/facebook-login/permissions#reference-default

Note: Available user properties vary from country to country

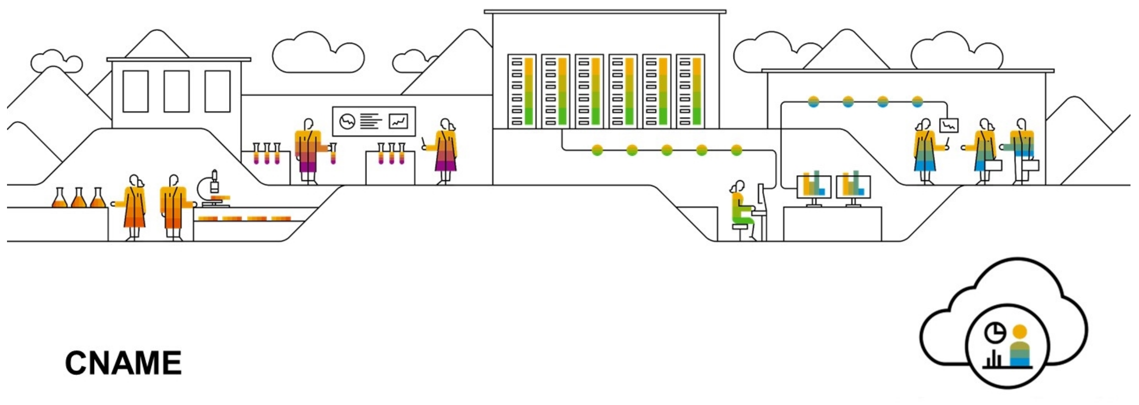
Figure 113: Basic Data Permissions

https://developers.facebook.com/docs/facebook-login/permissions#reference-default

- **Facebook**, **Instagram,** and **LinkedIn** *require approval* for extended data permissions
- To be approved to retrieve user data of extended permissions, you will need to go through a *submission process* with each of these social providers
- Social providers will require you to prove a use case for your website or application for each extended permission requested and for you to show how it benefits the end user's experience on your site or application

Figure 114: Extended Data Permissions

CNAME



CNAME

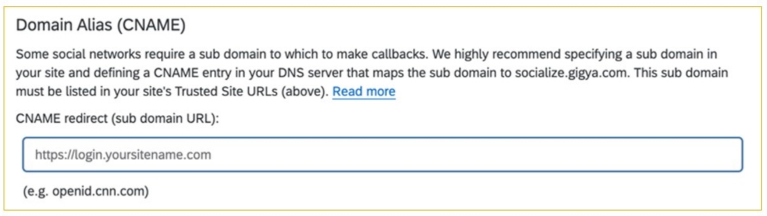
Figure 115: CNAME

- A Canonical Name or CNAME record is a type of DNS record that maps an alias name to a true or *canonical* domain name.
- CNAME records are typically used to map a subdomain such as *www* or *mail* to the domain hosting that subdomain's content.
- For example, a CNAME record can map the web address *www.example.com* to the actual web site for the domain *example.com*.
- In the example below, the subdomain that redirects to Customer Data Cloud will be mapped to socialize.gigya.com

https://login.yoursitename.com

Figure 116: CNAME

A Canonical Name or CNAME record is a type of DNS record that maps an alias name to a true or canonical domain name. CNAME records are typically used to map a subdomain such as www or mail to the domain hosting that subdomain's content. For example, a CNAME record can map the web address www.example.com to the actual web site for the domain example.com.



Domain Alias (CNAME)

Some social networks require a sub domain to which to make callbacks. We highly recommend specifying a sub domain in your site and defining a CNAME entry in your DNS server that maps the sub domain to socialize.gigya.com. This sub domain must be listed in your site's Trusted Site URLs (above). Read more

CNAME redirect (sub domain URL):

https://login.yoursitename.com

(e.g. openid.cnn.com)

- Not an actual landing page
- Only Customer Data Cloud can configure CNAME within the console
- This is required for staging and production environments, but not necessary for development environments

Figure 117: CNAME in Console

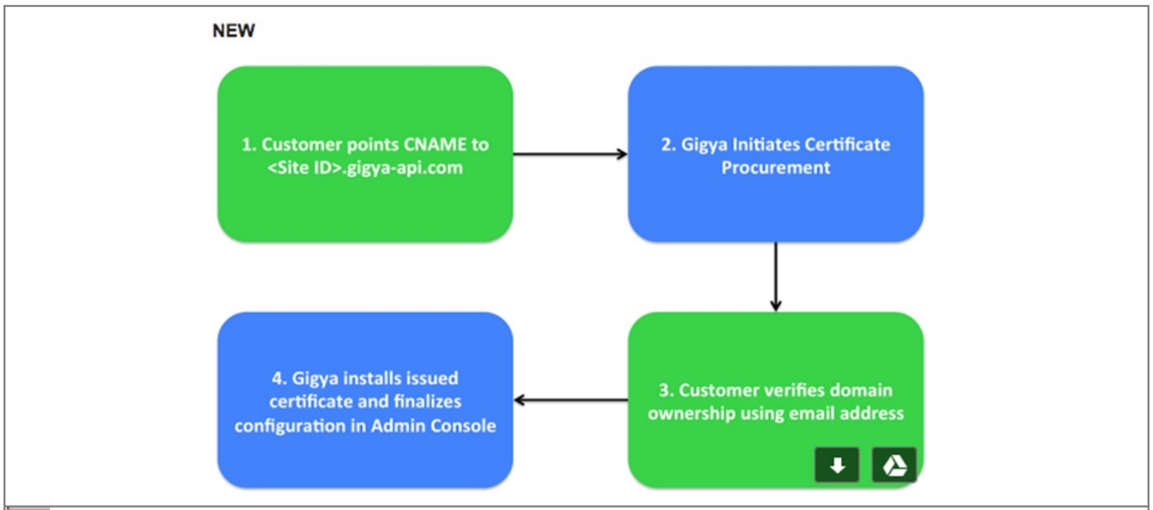

NEW
1. Customer points CNAME to <Site ID>.gigya-api.com
2. Gigya Initiates Certificate Procurement
3. Customer verifies domain ownership using email address
4. Gigya installs issued certificate and finalizes configuration in Admin Console

Figure 118: Configuring SSL Certificates - Existing CIAM Markets CN, RU

CIAM - Customer Identity and Access Management

In order to use a custom CNAME and avoid browser warnings related to 3rd party cookies, Customer Data Cloud can act as a client proxy for your site, thus eliminating browser security warnings. In order to do so, Customer Data Cloud uses an SSL certificate issued to your subdomain to encrypt your user's information transmitted during social login.

Note that this process is relevant to sites in the CN and RU data centers; The US, EU and AU data centers support an automated process for issuing an SSL certificate.

Why use an SSL Certificate?

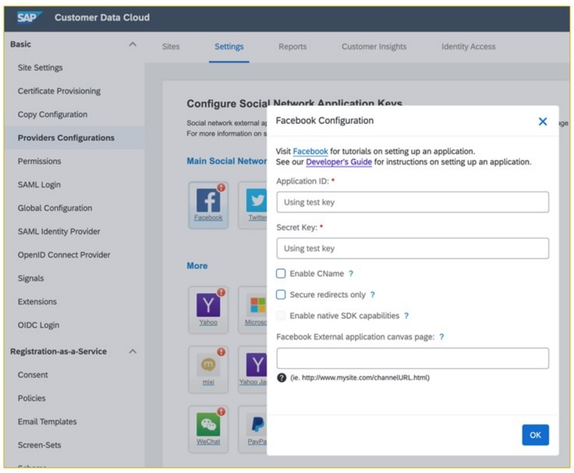
There are a number of reasons:

- Using a CNAME (and certificate) instead of socialize.gigya.com allows you to preserve the site branding and user experience that you've invested so much time and effort in.
- Because without one, every time your user is redirected from login.yoursite.com to Customer Data Cloud's servers, their browser will warn them with a security message. Using an SSL certificate verifies that the redirect to Customer Data Cloud's servers is a trusted process, thus eliminating this security popup for your site visitor's experience.
- Using an SSL certificate also ensures that data transmitted to and from the browser is encrypted, reinforcing trust between you and your visitors and helping to prevent middle-man attacks.

More info on setting up SSL for your site:

https://developers.gigya.com/display/GD/Site+Setup#SiteSetup-BeforeYouSetUpYourCertificate

https://developers.gigya.com/display/GD/Site+Setup#SiteSetup-ObtainingandInstallingtheCertificate



Facebook Configuration

Visit Facebook for tutorials on setting up an application.
See our Developer's Guide for instructions on setting up an application.

Application ID: *
Using test key

Secret Key: *
Using test key

☐ Enable CName ?
☐ Secure redirects only ?
☐ Enable native SDK capabilities ?

Facebook External application canvas page: ?

(ie. http://www.mysite.com/channelURL.html)

Figure 119: Social App Config

Some social networks require that callbacks go to a subdomain of your site and not directly to CDC . We highly recommend specifying a subdomain in your site and defining a CNAME entry in your DNS server that maps that subdomain to an alias you can receive from CDC support. CNAMEs provide the following benefits:

- Featuring your site in the OpenID authentication flow. Users will be prompted to allow your site (instead of allowing socialize.us1.gigya.com) to access the user's OpenID data.
- A better user experience on platforms such as iPhone, Android and Windows Mobile.

In the Admin console>> Site Settings >> Providers Configurations, you can configure different social network providers .

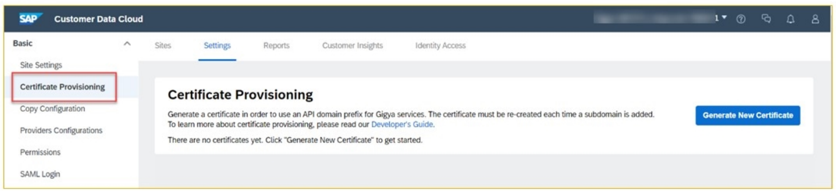
You can enable the CNAME but you need to first configure it.

Best practice is to mark "Secure redirects only" for all your applications. Soon CDC will only enable social network redirects through https protocol.

Certificate Provisioning

Figure 120: Certificate Provisioning

- Prevents browsers from blocking SAP Customer Data Cloud calls that are considered to be 3rd party calls
- Needed when configuring SSO for single domain or between multiple domains
- Every site will have a domain prefix with an SSL certificate connected to it
- It ensures ownership to all domains and subdomains via DNS verification



Certificate Provisioning

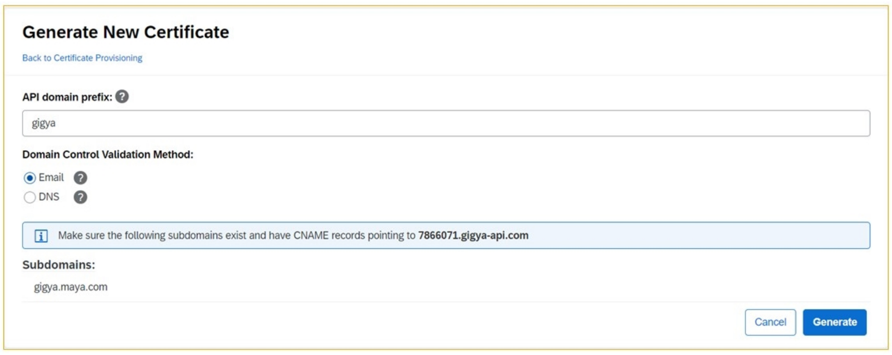
Generate a certificate in order to use an API domain prefix for Gigya services. The certificate must be re-created each time a subdomain is added.
To learn more about certificate provisioning, please read our Developer's Guide.

There are no certificates yet. Click "Generate New Certificate" to get started.

Figure 121: Certificate Provisioning

SSL certificate provisioning is only available for the US, EU, and AU data centers.

- Enter your API domain prefix, select your validation method (Email or DNS), double check your subdomains, and click Generate



Generate New Certificate

API domain prefix: gigya

Domain Control Validation Method:

[✓ Email] DNS

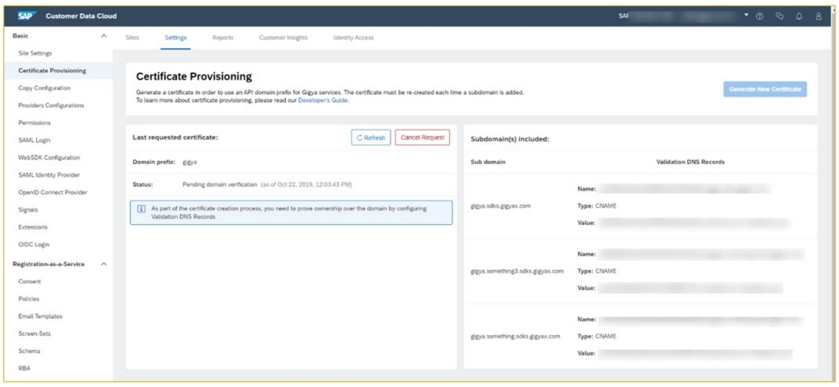
Make sure the following subdomains exist and have CNAME records pointing to 7866071.gigya-api.com

Subdomains:

gigya.maya.com

Figure 122: Generating a new Certificate

- Create the CNAME entries in your DNS configuration as described on the right panel



Certificate Provisioning

Generate a certificate in order to use an API domain prefix for Gigya services. The certificate must be re-created each time a subdomain is added.
To learn more about certificate provisioning, please read our Developer's Guide.

Last requested certificate:

Domain prefix: gigya

Status: Pending domain verification (as of Oct 22, 2019, 12:03:43 PM)

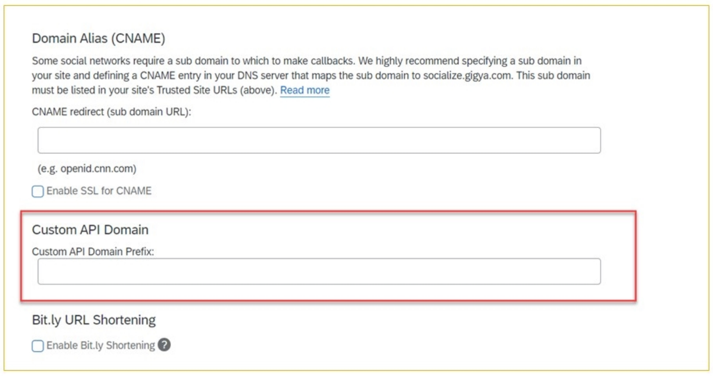
As part of the certificate creation process, you need to prove ownership over the domain by configuring Validation DNS Records

Subdomain(s) included:

Sub domain	Validation DNS Records
gigya.sdks.gigyax.com	Name: [redacted] Type: CNAME Value: [redacted]
gigya.something3.sdks.gigyax.com	Name: [redacted] Type: CNAME Value: [redacted]
gigya.something.sdks.gigyax.com	Name: [redacted] Type: CNAME Value: [redacted]

Figure 123: Verifying your domains

- Once the certificate is ready, go to Site Settings, fill in the Prefix in the Custom API Domain Prefix field, and click Save Settings.



Domain Alias (CNAME)

Some social networks require a sub domain to which to make callbacks. We highly recommend specifying a sub domain in your site and defining a CNAME entry in your DNS server that maps the sub domain to socialize.gigya.com. This sub domain must be listed in your site's Trusted Site URLs (above). Read more

CNAME redirect (sub domain URL):

(e.g. openid.cnn.com)

☐ Enable SSL for CNAME

Custom API Domain

Custom API Domain Prefix:

Bit.ly URL Shortening

☐ Enable Bit.ly Shortening

Figure 124: Finalizing the Certificate

Phone Number Login

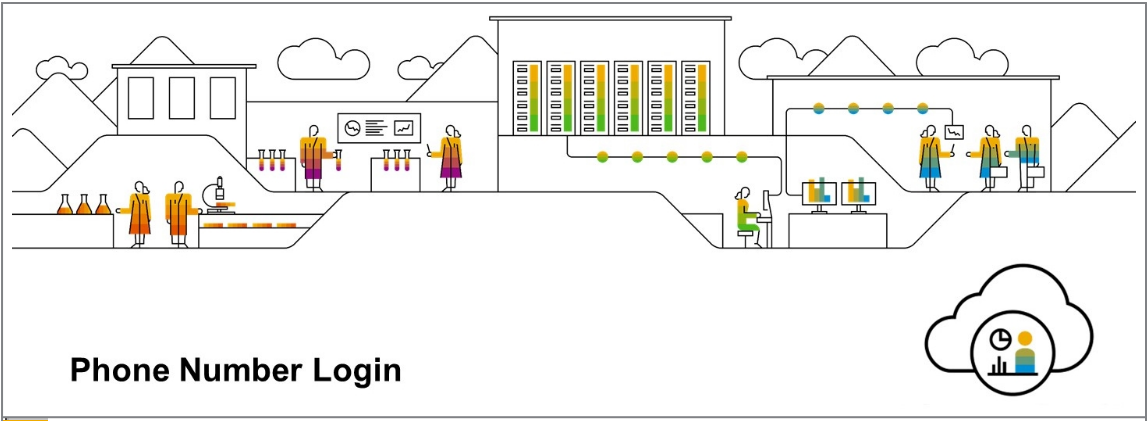
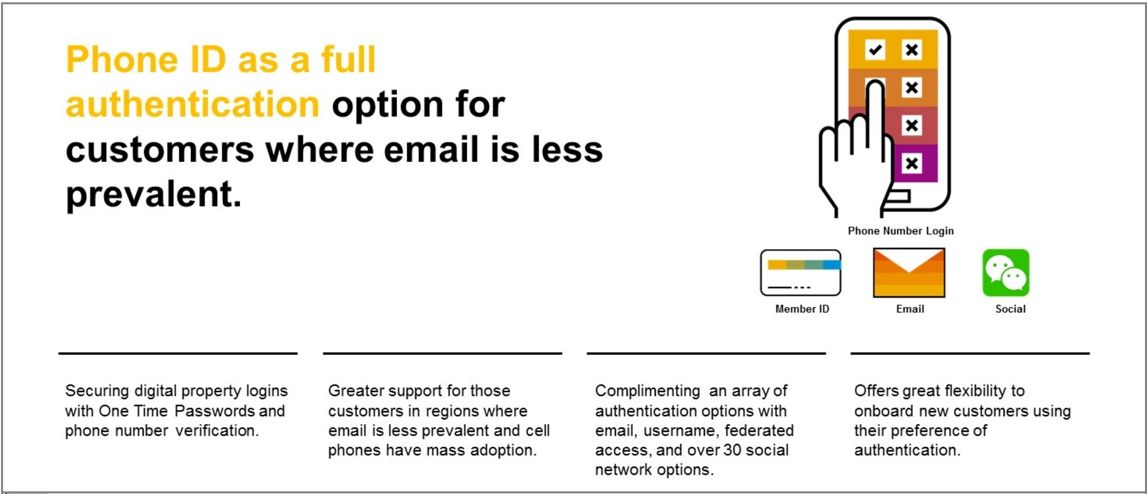


Figure 125: Phone Number Login



Phone ID as a full authentication option for customers where email is less prevalent.

Phone Number Login
Member ID
Email
Social

Securing digital property logins with One Time Passwords and phone number verification.

Greater support for those customers in regions where email is less prevalent and cell phones have mass adoption.

Complimenting an array of authentication options with email, username, federated access, and over 30 social network options.

Offers great flexibility to onboard new customers using their preference of authentication.

Figure 126: Phone ID Authentication - Business View and Benefits

Configure SMS providers like Twilio and LiveLink to send one time passwords.

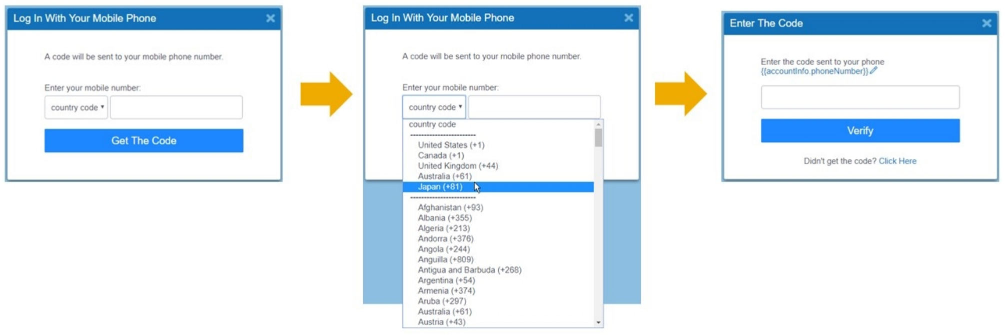
Manage the phone number blacklist and allow CSR's to search and update the phone number.

Pre-built screen-set support for Phone ID Authentication to improve implementation time and maintenance.

Login with a combination of Phone ID and One Time Password and lay the foundations for future authentication expansion.

Figure 127: Phone ID Authentication - Key Capabilities

1. The mobile login screens are included in the **RegistrationLogin** screen-set.
2. Select the **Mobile Login** or the **Mobile Login Verification** screens



A code will be sent to your mobile phone number.

Enter your mobile number:

Enter the code sent to your phone {{accountInfo.phoneNumber}}

Didn't get the code? Click Here

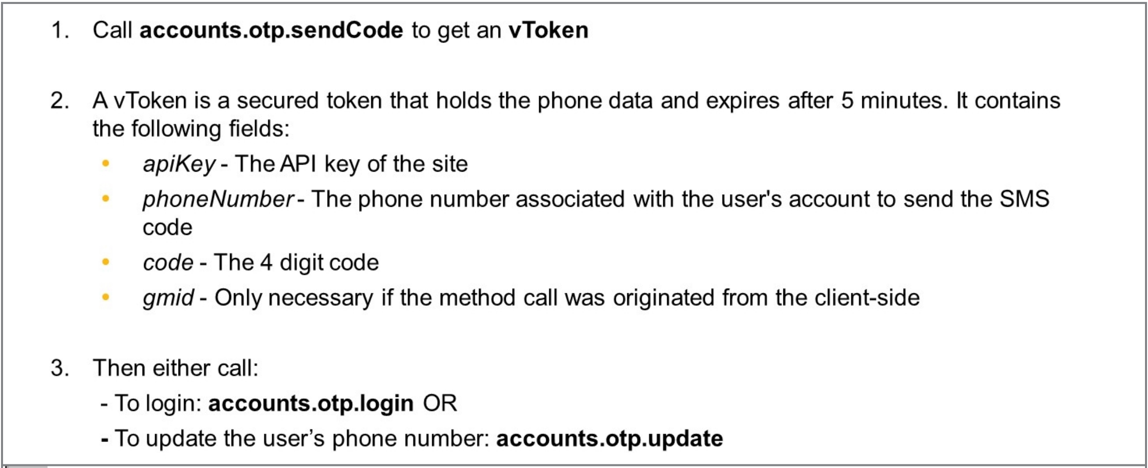
Figure 128: Phone ID Authentication - ScreenSets

Only the Mobile Login screens should be used for phone number login. No other screens will fully support this functionality.

1. Open the CDC console and go to the Screen-Sets page.
2. The mobile login screens are included in the RegistrationLogin screen-set. Open that screen-set to edit them.
3. Select the Mobile Login or the Mobile Login Verification screens on the left hand menu, under Screens.
4. You may change colors, assign CSS, and change the look and feel of the screen, but do not add any input fields. For more information on editing screens, check the UI Builder option.
5. Save your changes.

To gather consent from a user that is registering with phone number login (assuming you are using Enterprise Preference Management), all necessary consent statements must be on the Registration Completion screen.

If your site policy includes COPPA compliance, make sure to include the birth date fields in the Registration Completion screen.



1. Call **accounts.otp.sendCode** to get an **vToken**

2. A vToken is a secured token that holds the phone data and expires after 5 minutes. It contains the following fields:
   - *apiKey* - The API key of the site
   - *phoneNumber* - The phone number associated with the user's account to send the SMS code
   - *code* - The 4 digit code
   - *gmid* - Only necessary if the method call was originated from the client-side

3. Then either call:
   - To login: **accounts.otp.login** OR
   - To update the user's phone number: **accounts.otp.update**

Figure 129: Phone ID Authentication - REST API

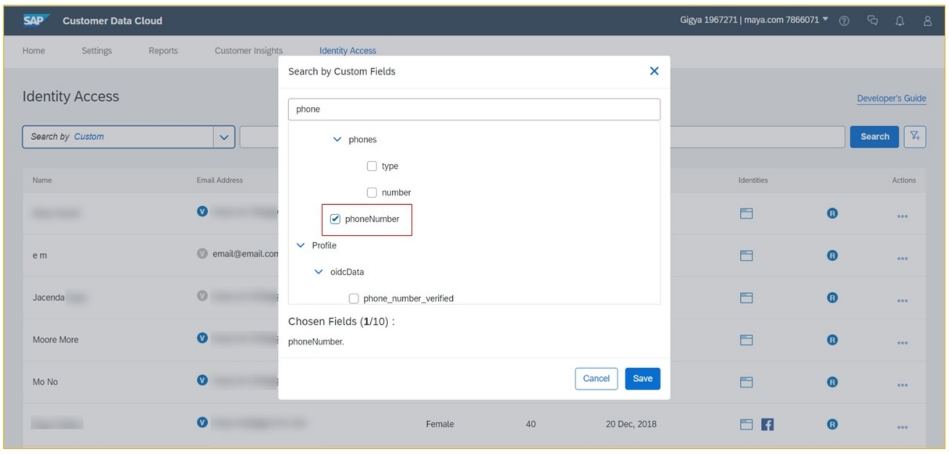
Either accounts.otp.login or accounts.otp.update require the OTP returned from accounts.otp.sendCode

More on these calls:

https://developers.gigya.com/display/GD/accounts.otp.sendCode+REST

https://developers.gigya.com/display/GD/accounts.otp.login+REST

https://developers.gigya.com/display/GD/accounts.otp.update+REST



Search by Custom Fields

phone

- phones
  - ☐ type
  - ☐ number
  - ☑ phoneNumber
- Profile
  - oidcData
    - ☐ phone_number_verified

Chosen Fields (1/10) :
phoneNumber.

Figure 130: Phone ID Authentication - Identity Access

You may search for accounts that have a value in the phoneNumber field, by calling accounts.search. For example: select * from accounts WHERE phoneNumber !=null

The audit log includes calls for accounts.otp.login and accounts.otp.sendCode

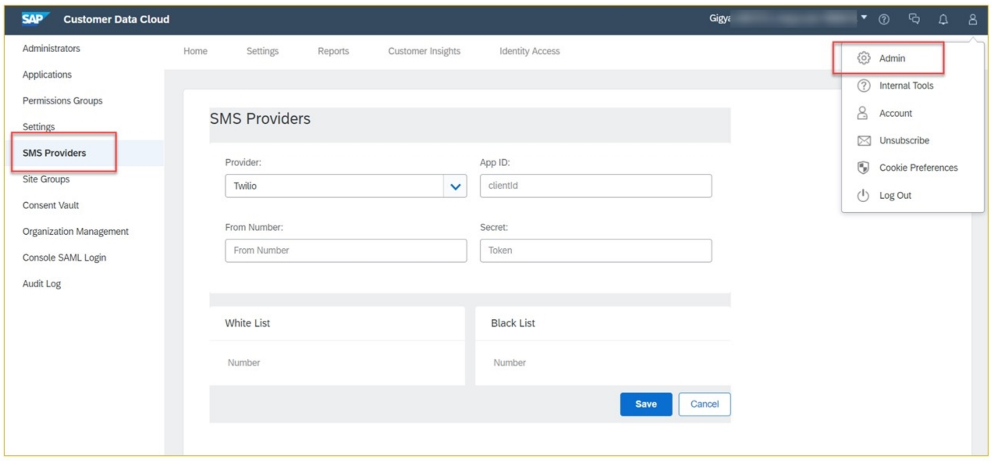


Figure 131: Phone ID Authentication - SMS Providers

SAP Customer Data Cloud needs an external SMS provider service in order to sent SMS texts.

Push Authentication

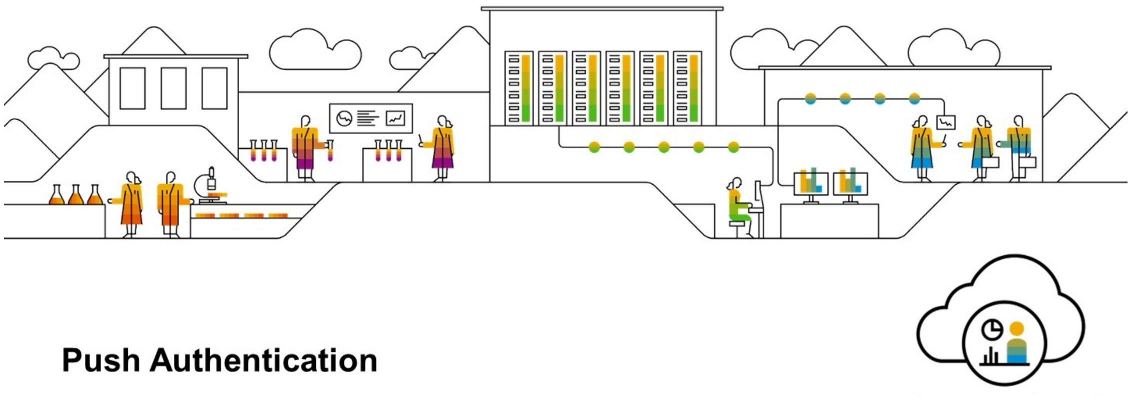


Figure 132: Push Authentication

- Passwordless login based on a push notification received on a device carrying a valid login session for the customer.

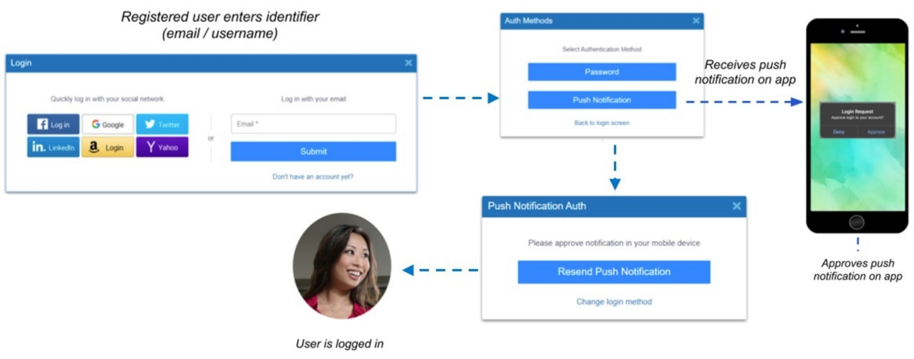
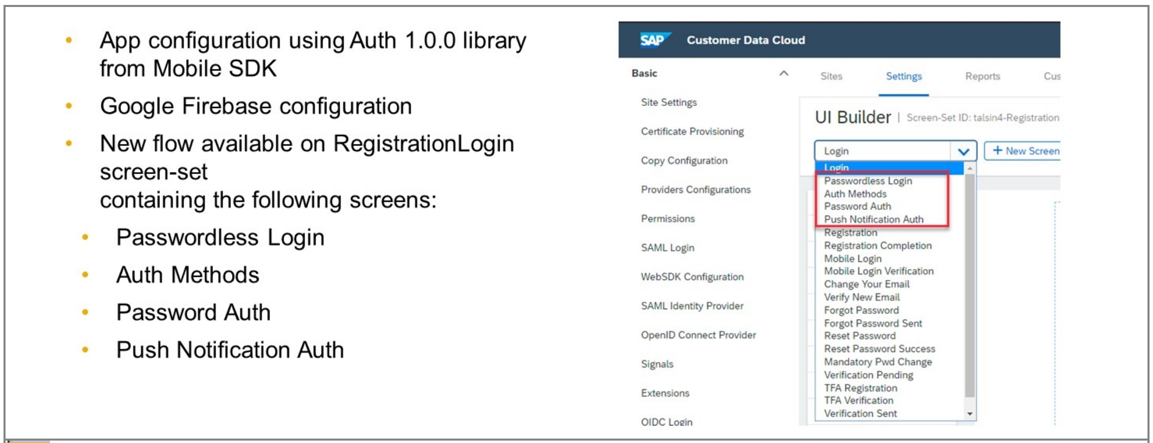

Registered user enters identifier (email / username)
Login
Quickly log in with your social network
Log in
Google
Twitter
LinkedIn
Login
Yahoo
or
Log in with your email
Email *
Submit
Don't have an account yet?
Auth Methods
Select Authentication Method
Password
Push Notification
Back to login screen
Receives push notification on app
Login Request
Approves push notification on app
Push Notification Auth
Please approve notification in your mobile device
Resend Push Notification
Change login method
User is logged in

Figure 133: Overview

The Push Authentication flow is as follows:

1. Customer registers to your website with email or username, creates a password (standard registration, not shown on graph above).
2. Customer logs into your app on their mobile phone, with the same identifier (email or username),
3. Customer opts for push notifications from inside the mobile app
4. The next time the customer logs in to your website, they can choose whether to authenticate with a password or a push notification.
5. If the customer chooses push notification, a notification is sent to their mobile phone.
6. After confirming the notification, the user is authenticated on your website.

For more please check https://developers.gigya.com/display/GD/Push+Authentication



- App configuration using Auth 1.0.0 library from Mobile SDK
- Google Firebase configuration
- New flow available on RegistrationLogin screen-set containing the following screens:
  - Passwordless Login
  - Auth Methods
  - Password Auth
  - Push Notification Auth

Figure 134: Implementation

App configuration is the most extensive step here, please follow instructions available at https://developers.gigya.com/display/GD/Push+Authentication#PushAuthentication-Implementation

The following relevant APIs are provided for Push Authentication functionality:

accounts.auth.getMethods

accounts.auth.push.sendVerification

accounts.auth.push.verify

accounts.auth.push.isVerified

accounts.auth.login

accounts.devices.register

accounts.devices.unregister

accounts.devices.update

Site Groups & Single Sign-On

Figure 135: Site Groups & Single Sign-On

- **Site Groups** – When you own multiple sites and wish to have a unified database of all users and a centralized place for setting configurations. A site group consists of exactly one parent site and one or more child sites.
- **Single Sign-On (SSO)** – Allows a user to be seamlessly signed in to a site after having signed in to another site.
- **SSO Segment** – Within a site group, you can define a sub-group of sites, known as SSO Segments, which share an SSO experience.

Figure 136: Key Terms

Parent sites and data centers cannot be changed at a later date.

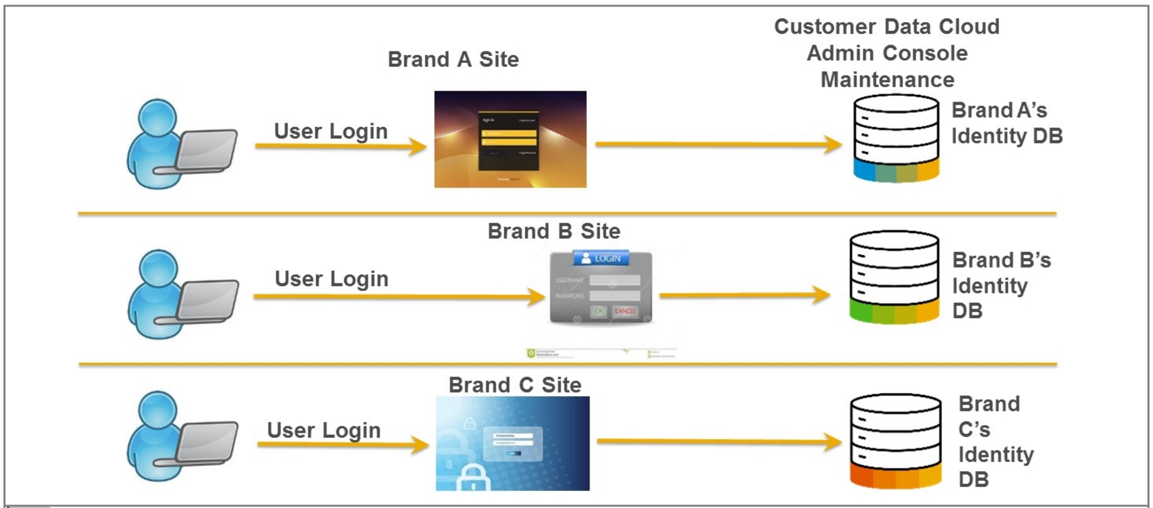


Figure 137: Without Site Groups

This is what we saw so far in previous chapters. Different single sites.

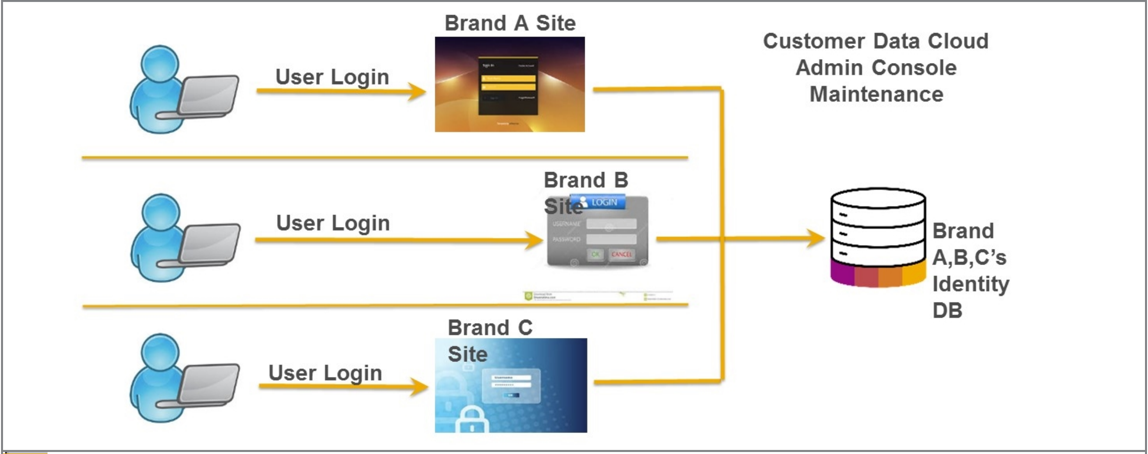

Brand A Site
User Login
Brand B Site
User Login
Brand C Site
User Login
Customer Data Cloud Admin Console Maintenance
Brand A,B,C's Identity DB

Figure 138: Site Groups (SSO Off)

Site Groups unify configurations and accounts of many children sites in a single parent site.

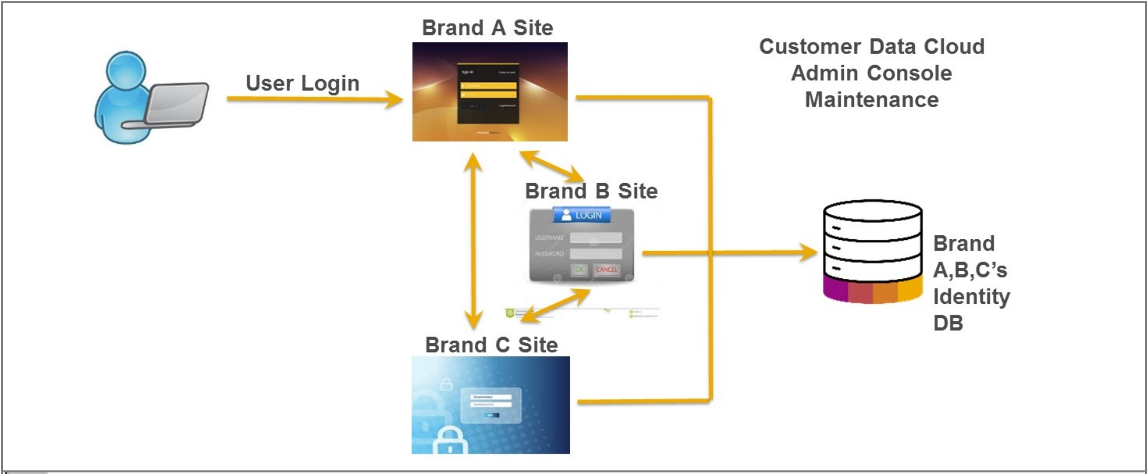

Brand A Site
User Login
Brand B Site
Brand C Site
Customer Data Cloud Admin Console Maintenance
Brand A,B,C's Identity DB

Figure 139: Site Groups (SSO On)

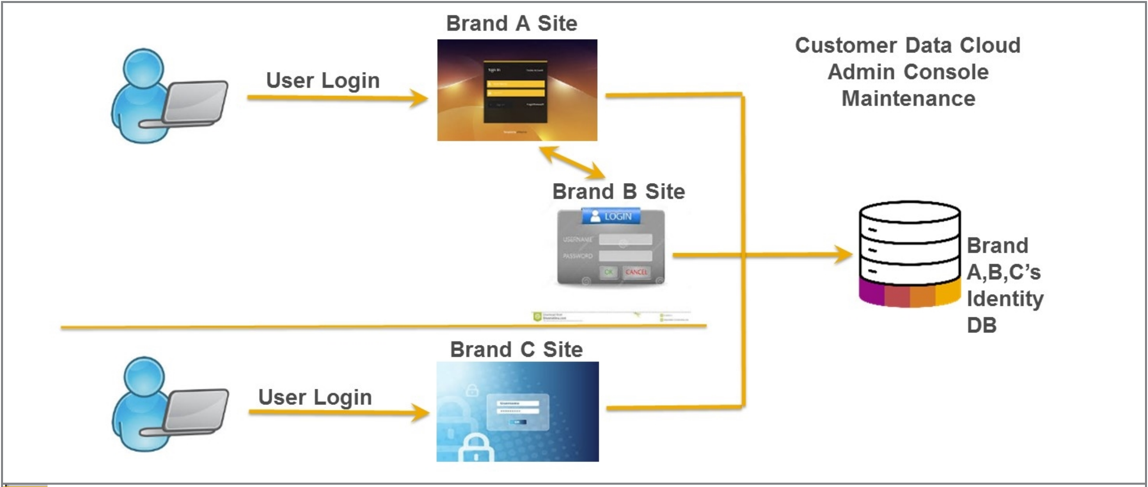
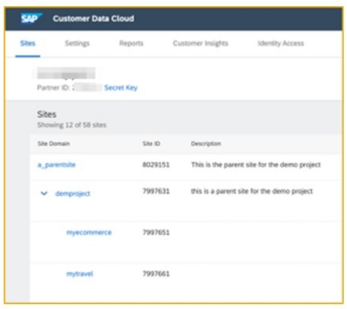

Brand A Site
User Login
Brand B Site
Brand C Site
User Login
Customer Data Cloud Admin Console Maintenance
Brand A,B,C's Identity DB

Figure 140: SSO Segments (Best of Both Worlds)

Brand A Site and Brand B Site are part of same SSO Segment.

SSO Segments - What are they?

- Within a site group, you can define a sub-group of sites, known as SSO Segments, which share an SSO experience
- SSO manages a single session for a user for all the sites in the segment, so that when a user is logged in to one site, and they visit another site in the group, they are logged in and gain immediate access to all other sites in that segment, saving them the trouble of authenticating separately to each site in the segment
- Any site can belong to only a single segment



[redacted]
Partner ID: [redacted] Secret Key

Sites
Showing 12 of 58 sites

Site Domain	Site ID	Description
a_parentsite	8029151	This is the parent site for the demo project
demproject	7997631	this is a parent site for the demo project
myecommerce	7997651	
mytravel	7997661	

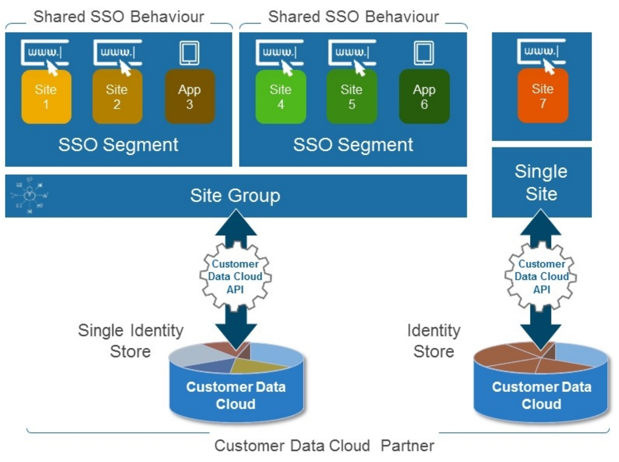


Figure 141: Customer Identity - Simple, Segmented Single Sign On (SSO)

Identity store = database

Scenario 1: one database common for all sites

Scenario 2: one database per site

1. By default, each API key (i.e. each site or app) has its own DB, siloed from other API keys
2. But API keys can be grouped under a master site, resulting in several API keys sharing user storage and settings
3. For example, once grouped, if I register on site 1, I can use the same credentials (and therefore login with the same account) on site 5
4. If the auto-login feature is activated on a site group, the session is maintained and shared between sites (if the sites are within same segment)

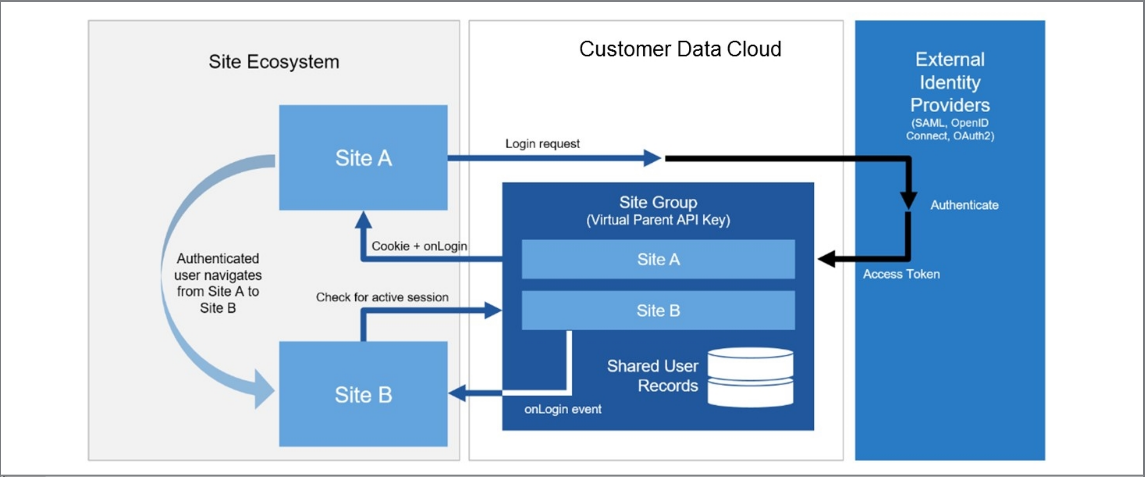
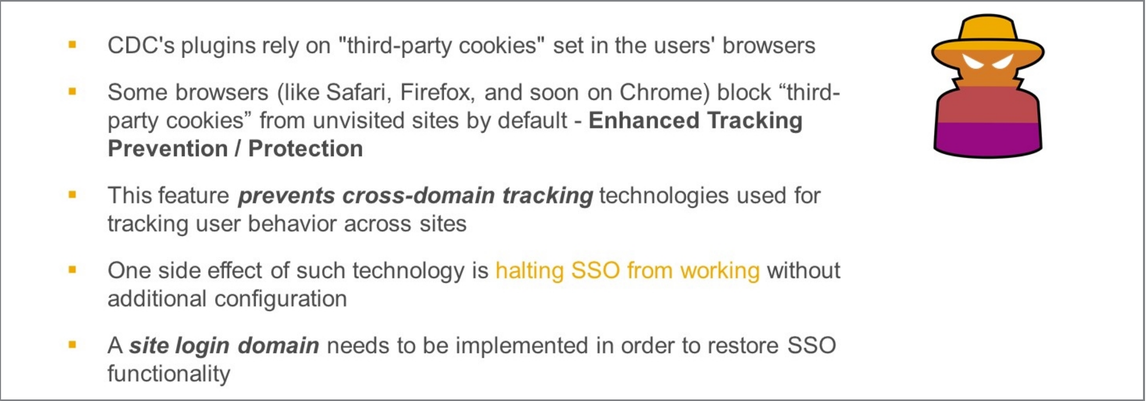


Figure 142: Sample Use Case: Single Sign-On Across Web Properties

SSO Important facts:

- You must use the same Data Center for each API key!
- If the client is unsure that they will use SSO, it is better to create an SSO group with their API keys because it is easier to disable them than to enable them once they have data on them. This would require an export followed by an import, which will waste project hours!



- CDC's plugins rely on "third-party cookies" set in the users' browsers
- Some browsers (like Safari, Firefox, and soon on Chrome) block “third-party cookies” from unvisited sites by default - **Enhanced Tracking Prevention / Protection**
- This feature ***prevents cross-domain tracking*** technologies used for tracking user behavior across sites
- One side effect of such technology is halting SSO from working without additional configuration
- A ***site login domain*** needs to be implemented in order to restore SSO functionality

Figure 143: Browser Track Prevention

SSO with Browser Tracking Prevention

https://developers.gigya.com/display/GD/Site+Groups+and+Single+Sign-On#SiteGroupsandSingleSign-On-SSOwithBrowserTrackingPrevention

Safari & Firefox have decided to completely block 3rd party cookies.

Incognito/private browsing mode has no effect on this behavior.

By default, when the Gigya Web SDK is loaded in a webpage, if it sees the user is using a browser that blocks third-party cookies by default (e.g., Safari), it will attempt to set the required data in Local Storage to solve the problem.

https://developers.gigya.com/display/GD/Blocked+Third-Party+Cookies

https://developers.gigya.com/display/GD/SSO+with+Safari

- To enable **single-domain SSO**
  - In the parent site's WebSDK Configuration, **storageDomainOverride** needs to be set to (custom API domain prefix + site domain), eg., login.example.com
  - A ***Custom Domain Prefix*** needs to point to your login domain
  - ***Certificate Provisioning*** configuration must be provided
- To enable **SSO on** all browsers for sites that use **different domains**
  - Set up a **centralized login domain**, to which all sites in the group will redirect their login requests.

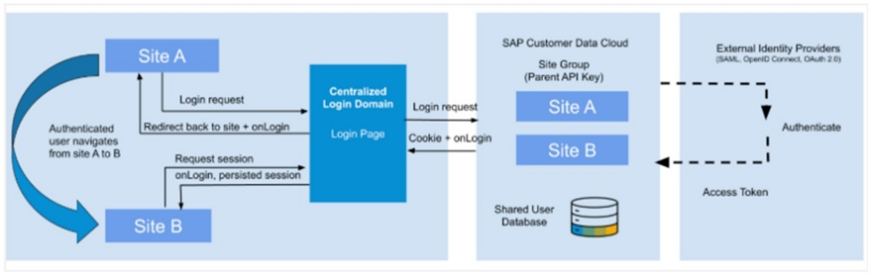


Figure 144: Centralized Site Login Domain

Browser Track Prevention:

- Some browsers (like Safari, Firefox, and soon to be available on Chrome as well) implement what is called Enhanced Tracking Prevention / Protection
- This feature prevents cross-domain tracking technologies used for tracking user behavior across sites
- One side effect of such technology is halting SSO from working without additional configuration
- A site login domain needs to be implemented in order to restore SSO functionality

Please see https://developers.gigya.com/display/GD/Site+Groups+and+Single+Sign-On#SiteGroupsandSingleSign-On-SSOwithBrowserTrackingPrevention

and https://developers.gigya.com/display/GD/Certificate+Provisioning

- On a site group, create a login redirect page and include CDC login functionality (Web SDK & Screen-sets)
  - *The CDC site hosting the Central Login Page should have the least restrictive policies*
  - The hosting CDC site would usually be the parent site of the site group

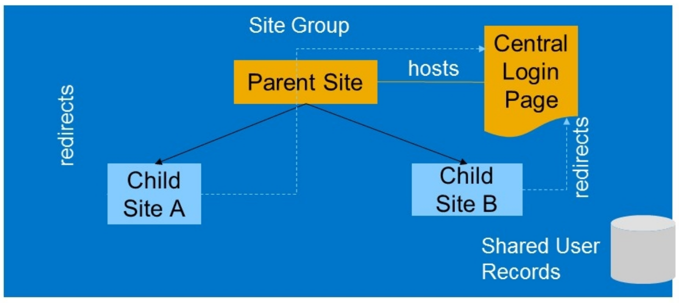


Figure 145: Best Practice of Central Login Page

Parent and Child Scopes

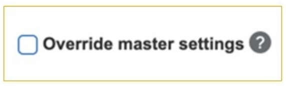
Parent and Child Scopes

Figure 146: Parent and Child Scopes

By default, site settings for all the sites in a Site Group are controlled by the **parent site**

A **child site** may override the following master settings:

- Data Schema
- Consent
- Site Policies
- Permissions
- Screen-Sets
- Email Templates
- Reports
- WebSDK Configurations



☐ **Override master settings** ?

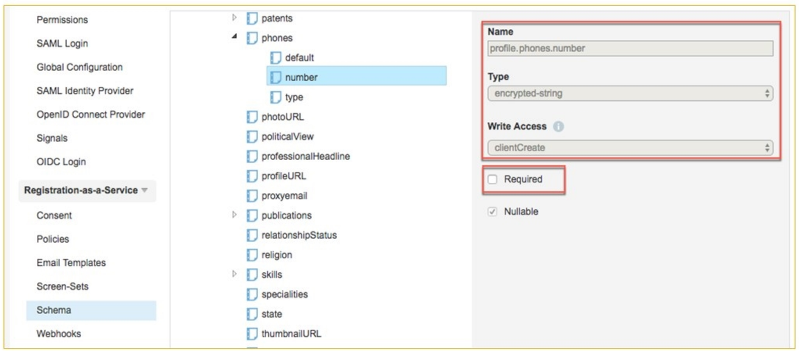
Figure 147: Parent and Child Scopes

https://developers.gigya.com/display/GD/Site+Groups+and+Single+Sign-On

https://developers.gigya.com/display/GD/accounts.setPolicies+REST#accounts.setPoliciesREST-OverridingMasterConfigurations

All sites that use Customer Identity and are part of the same site group must contain ***identical data schemas*** as all sites of a site group use **the same data base**

A **child site** may alter a field's *Required* attribute



Name: profile.phones.number

Type: encrypted-string

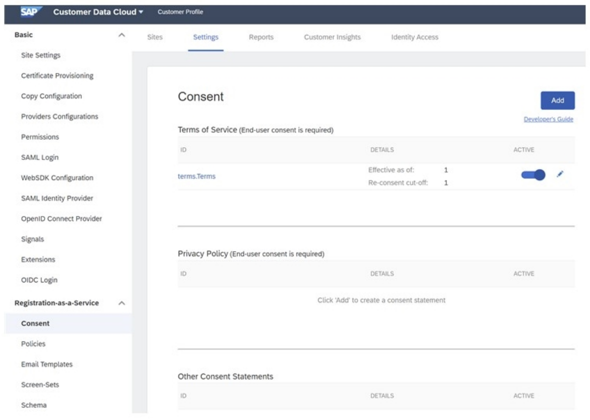
Write Access: clientCreate

☐ Required

☑ Nullable

Figure 148: Data Schemas

- In a **child site** of a site group, you can now reset the consent statement status for a user by calling ***accounts.resetSitePreferences***
- This method will reset only consent statements that are unique to the site, i.e. will not affect statements that are active in other sites of the group



Consent

Terms of Service (End-user consent is required)

ID	DETAILS	ACTIVE
terms.Terms	Effective as of: 1 Re-consent cut-off: 1	

Privacy Policy (End-user consent is required)

ID	DETAILS	ACTIVE

Click 'Add' to create a consent statement

Other Consent Statements

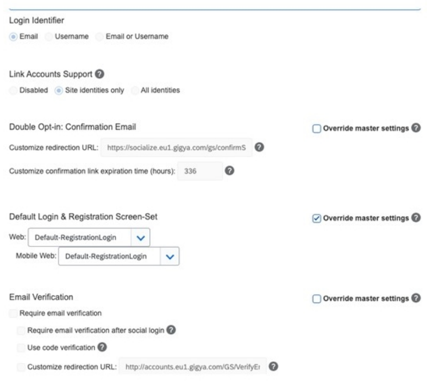
ID	DETAILS	ACTIVE

Figure 149: Consent

This method is used to reset the preferences (consent statement status) of a registered site user, on a child site in a site group, for all the consent statements including any mandatory consent statements to which they have agreed.

https://developers.gigya.com/display/GD/accounts.resetSitePreferences+REST

- Most site policies are set on the pa**rent site** and affect the entire group
- There is an Override master setting checkbox next to the relevant configuration
- A few policies can be overwritten on the c**hild site**:



Login Identifier
[✓ Email] Username Email or Username

Link Accounts Support
Disabled [✓ Site identities only] All identities

Double Opt-in: Confirmation Email ☐ Override master settings
Customize redirection URL: https://socialize.eu1.gigya.com/gs/confirmS
Customize confirmation link expiration time (hours): 336

Default Login & Registration Screen-Set ☑ Override master settings
Web: Default-RegistrationLogin
Mobile Web: Default-RegistrationLogin

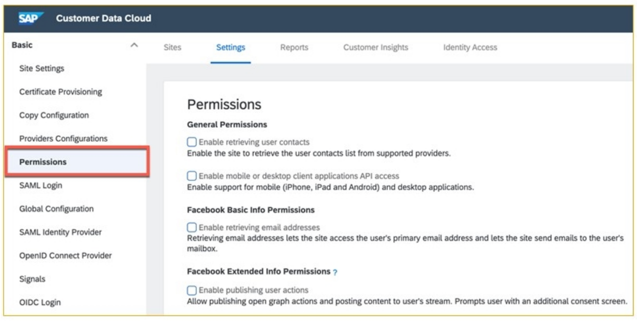
Email Verification ☐ Override master settings
☐ Require email verification
☐ Require email verification after social login
☐ Use code verification
☐ Customize redirection URL: http://accounts.eu1.gigya.com/GS/VerifyEı

- **verifyEmail** object
  (of the **AccountOptions** parameter)
- **verifyProviderEmail** object
- **emailVerification**
- **emailNotifications**
- **passwordReset**
- **requireCaptcha**

Figure 150: Site Policies

.

- Permissions can be unique per site or shared among all sites, depending on the number of social network apps defined in the group
- When using a single app for multiple websites, permissions for all sites must be identical and defined at both the parent and child level



Permissions

General Permissions

☐ Enable retrieving user contacts
Enable the site to retrieve the user contacts list from supported providers.

☐ Enable mobile or desktop client applications API access
Enable support for mobile (iPhone, iPad and Android) and desktop applications.

Facebook Basic Info Permissions

☐ Enable retrieving email addresses
Retrieving email addresses lets the site access the user's primary email address and lets the site send emails to the user's mailbox.

Facebook Extended Info Permissions ?

☐ Enable publishing user actions
Allow publishing open graph actions and posting content to user's stream. Prompts user with an additional consent screen.

Figure 151: Permissions

- By default, all of a group's child sites share the parent's screen-sets
- Child sites can have unique screen-sets
- If both child and parent have the same screen-set name, the child screen-set always overrides the parent's

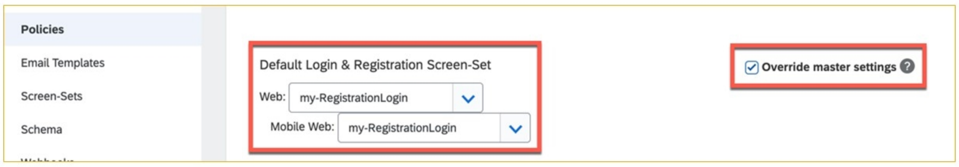
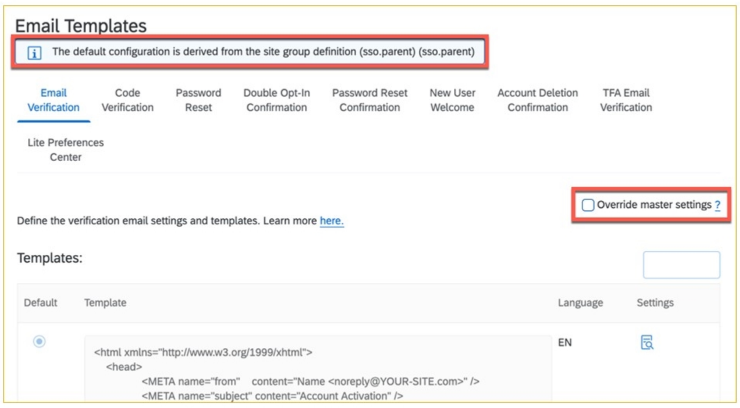

Policies
Email Templates
Screen-Sets
Schema
Default Login & Registration Screen-Set
Web: my-RegistrationLogin
Mobile Web: my-RegistrationLogin
Override master settings

Figure 152: Screen-Sets

- By default, group sites share the same email templates
- However, a child site can have its own email template


Email Templates

The default configuration is derived from the site group definition (sso.parent) (sso.parent)

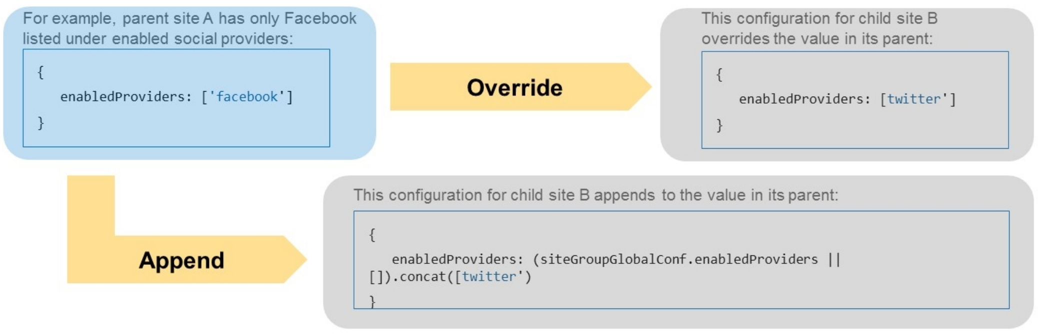
Define the verification email settings and templates. Learn more here.

Templates:

Default	Template	Language	Settings
●	<html xmlns="http://www.w3.org/1999/xhtml"> <head> <META name="from" content="Name <noreply@YOUR-SITE.com>" /> <META name="subject" content="Account Activation" />	EN	

Figure 153: Email Templates

- The WebSDK configuration of the parent is inherited by the child sites
- A child parameter overrides the parameter defined in the parent WebSDK config
- Use custom code to have the settings of the child site appended to those of the parent


For example, parent site A has only Facebook listed under enabled social providers:

```
{
   enabledProviders: ['facebook']
}
```

This configuration for child site B overrides the value in its parent:

```
{
   enabledProviders: [twitter']
}
```

This configuration for child site B appends to the value in its parent:

```
{
   enabledProviders: (siteGroupGlobalConf.enabledProviders ||
[]).concat([twitter'])
}
```

Figure 154: WebSDK Configuration in Site Group Scenarios

Global Access

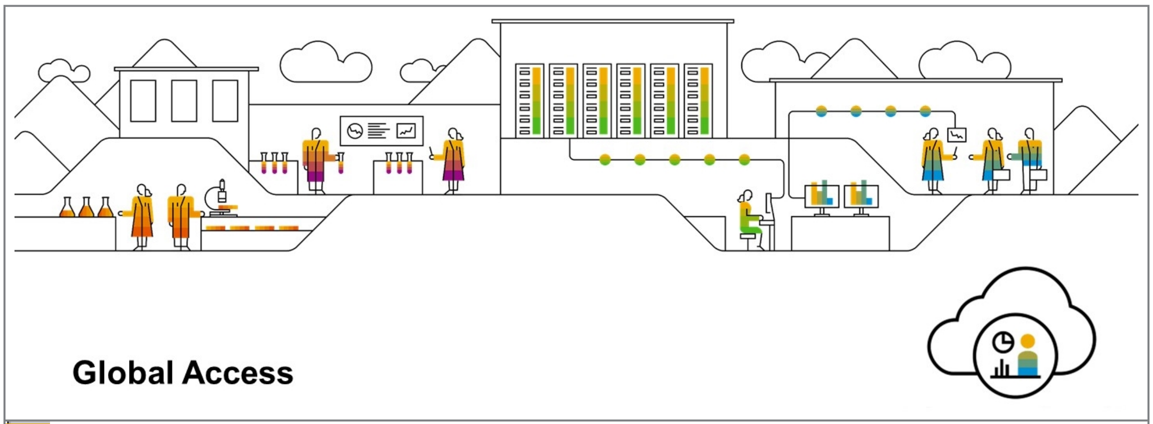


Figure 155: Global Access

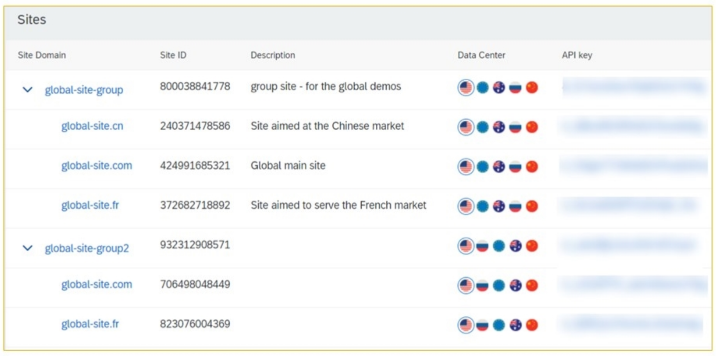
- Gives customers a consistent user experience, regardless of their physical location
- User data is stored at the data center they used to create their account
- Data residency regulations are upheld without compromising experience
- Based on storing user data for a site or site group in different data centers
- A single user database with users from multiple regions

Figure 156: Key Features

Refer to this link for more details:

https://developers.gigya.com/display/GD/Global+Access

- Site configurations in all the site data centers: site settings, schema, policies, permission groups, user keys, and screen-sets, are replicated
- User database is never replicated between data centers
- This applies to both Single Global Sites and Global Site Groups



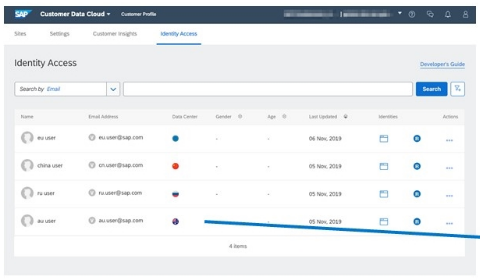
Sites

Site Domain	Site ID	Description	Data Center	API key
global-site-group	800038841778	group site - for the global demos	🇺🇸 🇪🇺 🇦🇺 🇷🇺 🇨🇳	[redacted]
global-site.cn	240371478586	Site aimed at the Chinese market	🇺🇸 🇪🇺 🇦🇺 🇷🇺 🇨🇳	[redacted]
global-site.com	424991685321	Global main site	🇺🇸 🇪🇺 🇦🇺 🇷🇺 🇨🇳	[redacted]
global-site.fr	372682718892	Site aimed to serve the French market	🇺🇸 🇪🇺 🇦🇺 🇷🇺 🇨🇳	[redacted]
global-site-group2	932312908571		🇺🇸 🇷🇺 🇪🇺 🇦🇺 🇨🇳	[redacted]
global-site.com	706498048449		🇺🇸 🇷🇺 🇪🇺 🇦🇺 🇨🇳	[redacted]
global-site.fr	823076004369		🇺🇸 🇷🇺 🇪🇺 🇦🇺 🇨🇳	[redacted]

Figure 157: Global Site Accounts and Configurations Data Residency

Global API keys are relatively short and start with 4_

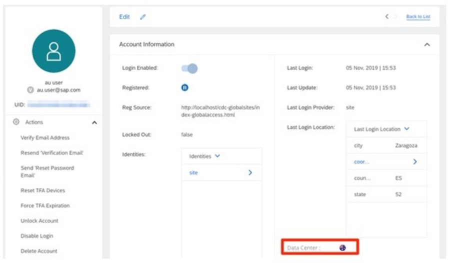
When Global Access is enabled for your partner, Identity Access includes a new "Data Center" column that displays the relevant flag.



Identity Access

Name	Email Address	Data Center	Gender	Age	Last Updated	Identities		Actions
eu user	eu.user@sap.com	●	-	-	06 Nov, 2019			...
china user	cn.user@sap.com	●	-	-	05 Nov, 2019			...
ru user	ru.user@sap.com	●	-	-	05 Nov, 2019			...
au user	au.user@sap.com	●	-	-	05 Nov, 2019			...

4 items



au user
au.user@sap.com
UID: [redacted]

Account Information

Login Enabled:
Registered:
Reg Source: http://localhost/cdc-globalsites/index-globalaccess.html
Locked Out: false
Identities: Identities — site
Last Login: 05 Nov, 2019 | 15:53
Last Update: 05 Nov, 2019 | 15:53
Last Login Provider: site
Last Login Location: Last Login Location

city	Zaragoza
coor...	
coun...	ES
state	52

Data Center :

The user record also shows the Data Center

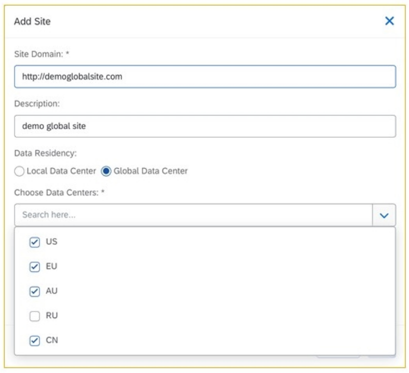
Figure 158: Account Data Residency

With the API method accounts.global.changeAccountResidency you can now change the resident data center of a user.

https://developers.gigya.com/display/GD/accounts.global.changeAccountResidency+REST

- Enter the Site Domain
- Select Global Data Center for Data Residency
- Select the data centers to be included in the site

- Primary datacenter should be the one that holds the majority of the user database and traffic for this global site or site group



Add Site

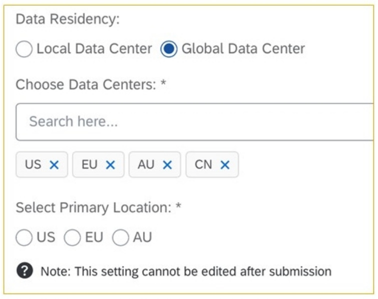
Site Domain: *
http://demoglobalsite.com

Description:
demo global site

Data Residency:
Local Data Center [✓ Global Data Center]

Choose Data Centers: *
Search here...

- [x] US
- [x] EU
- [x] AU
- [ ] RU
- [x] CN



Data Residency:
Local Data Center [✓ Global Data Center]

Choose Data Centers: *
Search here...
US ✕ EU ✕ AU ✕ CN ✕

Select Primary Location: *
US EU AU

Note: This setting cannot be edited after submission

Figure 159: Create a Global Site

- Global Access only supports following validation types

✓ Validate the Web SDK response using an id_token
✖ UIDSignature not supported

✓ Sign requests using a bearer token
✖ Signing with a user or application key not supported

Figure 160: Making API Calls

Validate A JWT from SAP Customer Data Cloud:

https://developers.gigya.com/display/GD/Validate+A+JWT+from+SAP+Customer+Data+Cloud

Signing Requests to SAP Customer Data Cloud:

https://developers.gigya.com/display/GD/Signing+Requests+to+SAP+Customer+Data+Cloud

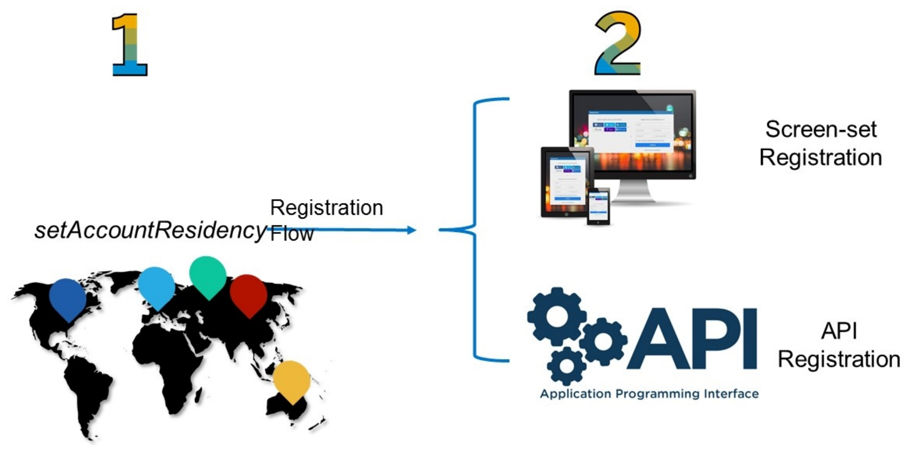


Figure 161: Global Registration Flow

gigya.setAccountResidency(params)

us1

eu1

au1

ru1

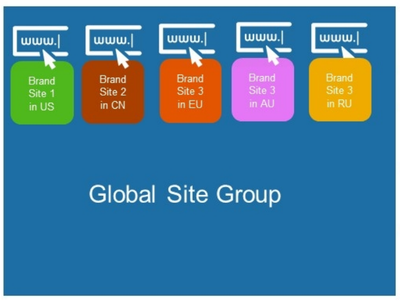
cn1

https://developers.gigya.com/display/GD/setAccountResidency+JS

API Registration:

accounts.initRegistration

socialize.login



- Can contain sites from any of the SAP CDC data centers
- Can belong to more than one data center
- Allows users to log in to all the sites in the group from different locations
- Can provide global users with a single sign-on experience
- All sites in a global site group must share the same data centers and the same primary data center

Figure 162: Global Site Group

Unsupported Features (as of July 2020):

- Link accounts flow
- Phone Number Login
- Lite Accounts

- Federation (SAML, OIDC)
- Limited social provider support: Apple, Facebook, Google, Twitter and WeChat are supported.
- CIAM for B2B
- Client-side calls of accounts.search

Please check https://developers.gigya.com/display/GD/Global+Access#GlobalAccess-UnsupportedFeatures for an updated list

Risk-Based Authentication

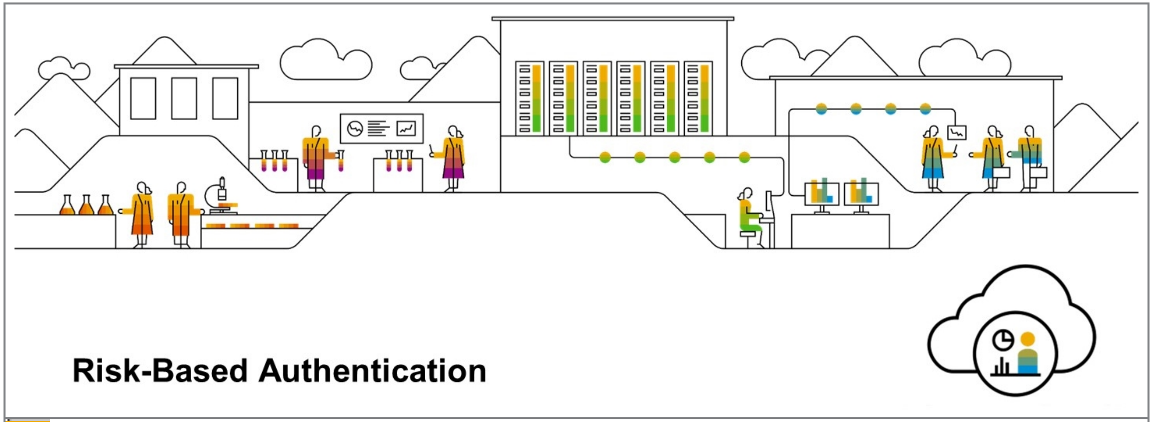
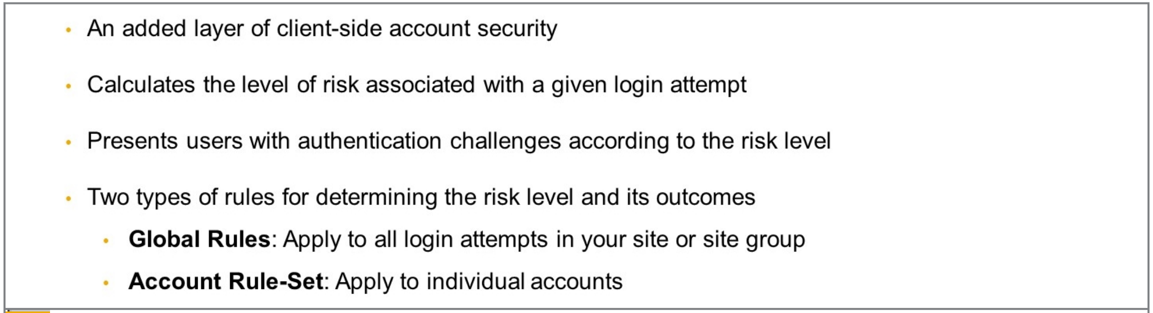


Figure 163: Risk-Based Authentication

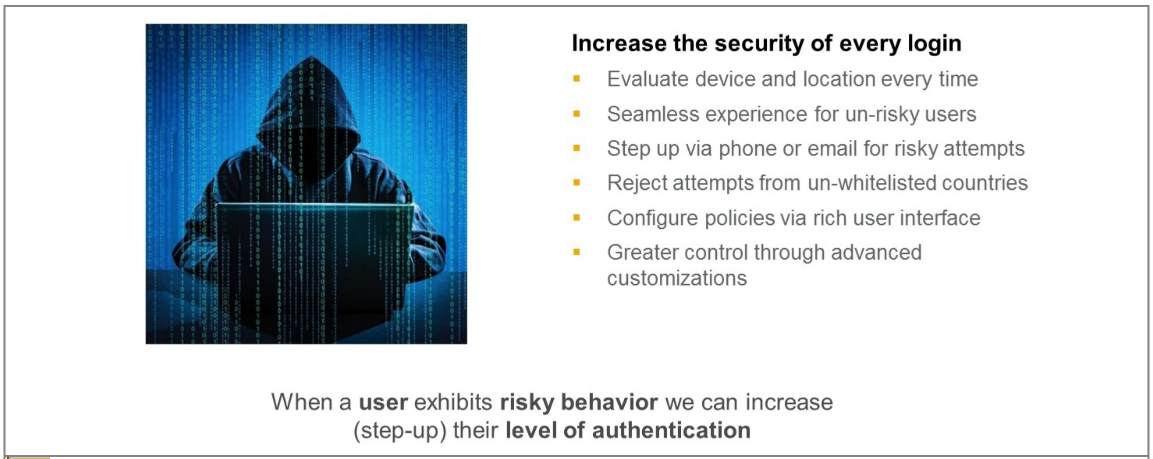


- An added layer of client-side account security
- Calculates the level of risk associated with a given login attempt
- Presents users with authentication challenges according to the risk level
- Two types of rules for determining the risk level and its outcomes
  - **Global Rules**: Apply to all login attempts in your site or site group
  - **Account Rule-Set**: Apply to individual accounts

Figure 164: Key Features

Refer to this video for more details:

https://enable.cx.sap.com/media/1_2r1zmjn3



Increase the security of every login

- Evaluate device and location every time
- Seamless experience for un-risky users
- Step up via phone or email for risky attempts
- Reject attempts from un-whitelisted countries
- Configure policies via rich user interface
- Greater control through advanced customizations

When a **user** exhibits **risky behavior** we can increase (step-up) their **level of authentication**

Figure 165: What is Risk-Based Authentication

If a user comes in from a different location or device, we can deem that as risky behavior, and require them to perform additional authentication. This is something we could always do programmatically, but now organizations are able to set this up using the UI within the product. Through the console UI (as opposed to the REST API), they can define what the risky behavior is, what type of authentication will be required and to which users this should apply.

- We can evaluate a user's device and location every time the user accesses the customer's digital property.
- If the user does not exhibit any risky behavior, there is NO CHANGE in the experience for that user.
- If the user does exhibit risky behavior (which can be configured by an organization), an extra authentication step (step-up authentication) can be added, such as phone or email verification.
- Organizations can reject login attempts from countries that they deem to be of concern.
- This can now all be configured through an easy-to-use user interface in the Customer Data Cloud Admin console.

We added Risk-Based Authentication late in 2016. However, there was no UI associated with it, so it could only be used by organizations who were using our Global Services team to implement the solution. With this release, we provided a UI capability to allow any organization to implement RBA

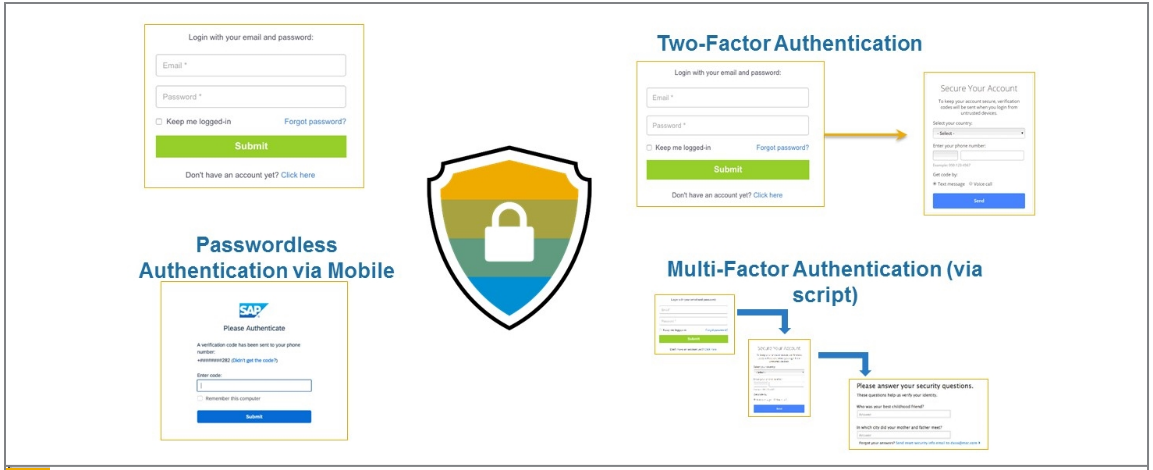

Two-Factor Authentication
Passwordless Authentication via Mobile
Multi-Factor Authentication (via script)
Login with your email and password:
Email *
Password *
Keep me logged-in
Forgot password?
Submit
Don't have an account yet? Click here
Secure Your Account
Please Authenticate
Please answer your security questions.

Figure 166: RBA Security Options

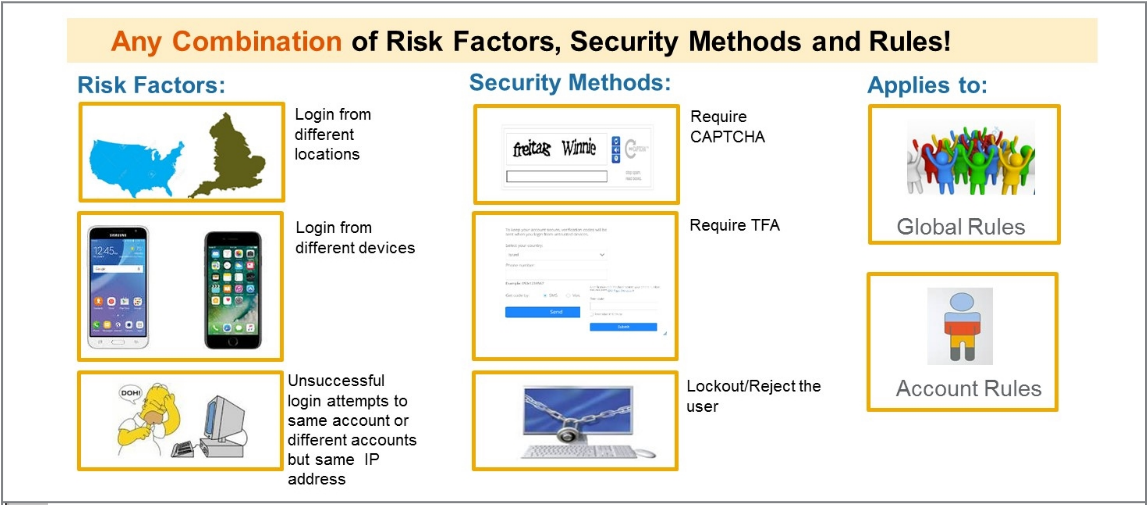
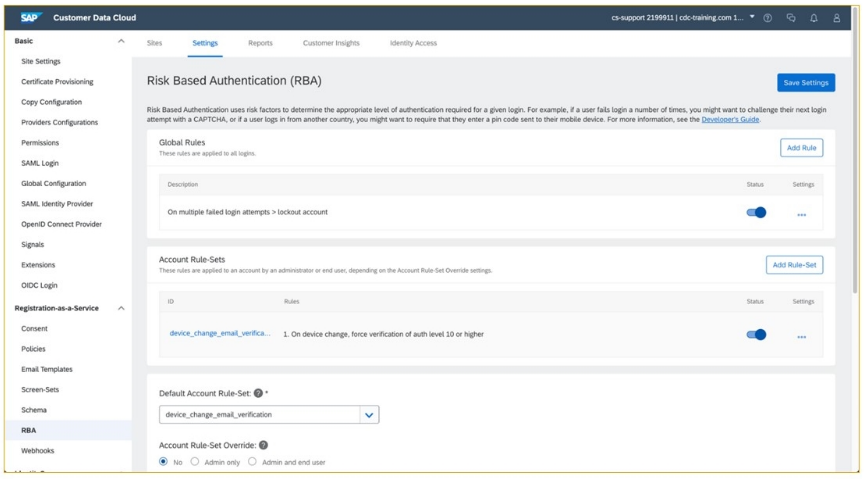

Any Combination of Risk Factors, Security Methods and Rules!
Risk Factors:
Login from different locations
Login from different devices
Unsuccessful login attempts to same account or different accounts but same IP address
Security Methods:
Require CAPTCHA
Require TFA
Lockout/Reject the user
Applies to:
Global Rules
Account Rules

Figure 167: RBA UI - Risk Factors > Security Methods > Selection Rules

When looking at risky behavior, we refer to that as risk factors in the RBA UI enhancement. Organizations can set up factors to look for in their community. These risk factors can be based on logons from a different country, so if a user logs in from the US today and UK tomorrow, we can dictate that as risky. Risk factors can also be based on a new device: if they login via Android one day and iPhone the next, that could be suspicious. Lastly, repeated unsuccessful login attempts to the same account or to different accounts from the same IP address should raise a red flag. You can choose all of these items, or a subset of them.

If the system experiences one or more of these risky behaviours, what can we do?



Risk Based Authentication (RBA)

Risk Based Authentication uses risk factors to determine the appropriate level of authentication required for a given login. For example, if a user fails login a number of times, you might want to challenge their next login attempt with a CAPTCHA, or if a user logs in from another country, you might want to require that they enter a pin code sent to their mobile device. For more information, see the Developer's Guide.

Global Rules
These rules are applied to all logins.

Description	Status	Settings
On multiple failed login attempts > lockout account	●	...

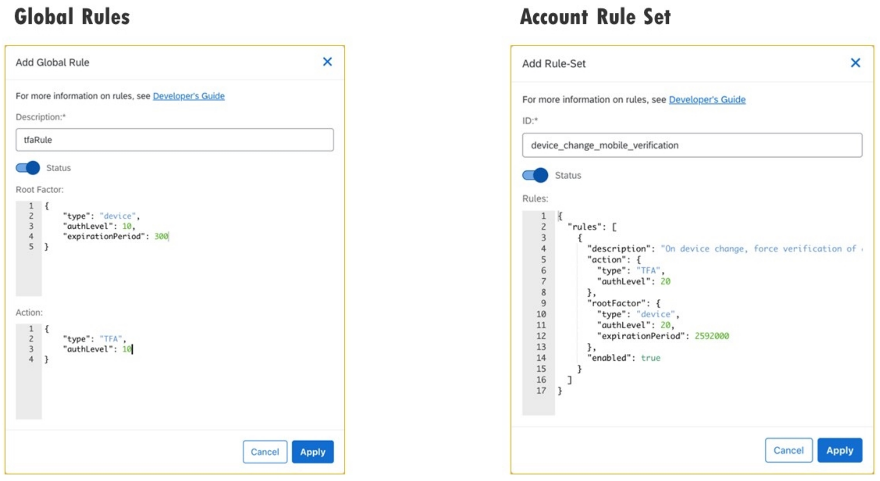
Account Rule-Sets
These rules are applied to an account by an administrator or end user, depending on the Account Rule-Set Override settings.

ID	Rules	Status	Settings
device_change_email_verifica...	1. On device change, force verification of auth level 10 or higher	●	...

Default Account Rule-Set: device_change_email_verification

Account Rule-Set Override: [✓ No] Admin only Admin and end user

Figure 168: RBA UI : Editor and Templates to make it easy to configure



Global Rules

Add Global Rule

For more information on rules, see Developer's Guide

Description:* tfaRule

Status

Root Factor:

```
{
    "type": "device",
    "authLevel": 10,
    "expirationPeriod": 300
}
```

Action:

```
{
    "type": "TFA",
    "authLevel": 10
}
```

Account Rule Set

Add Rule-Set

For more information on rules, see Developer's Guide

ID:* device_change_mobile_verification

Status

Rules:

```
{
  "rules": [
    {
      "description": "On device change, force verification of 
      "action": {
        "type": "TFA",
        "authLevel": 20
      },
      "rootFactor": {
        "type": "device",
        "authLevel": 20,
        "expirationPeriod": 2592000
      },
      "enabled": true
    }
  ]
}
```

Figure 169: Rules

Global rules are applied to all logins. You can find them under Admin Console >> Registration-as-a-Service >> RBA.

In this example, we are looking at a rule to apply two-factor authentication. You can base the new rule on the existing templates, or create a custom rule. A template-based rule can be edited in the next step.

For editing rules, visit https://developers.gigya.com/display/GD/Accounts+RBA+Policy+Object to comply with defined standards.

In the root factor, we are setting an expiration period for an authenticated user:

Auth level 10: email

Auth level 20: mobile

Action: If the root factor is true then an action takes place, which means the user is asked for 2-factor authentication

Setting up global rules:

- the risk factor is defined under Root Factor,
- the action taken if that factor is triggered, is defined under Action.

You can edit the following:

Description

Status: can be switched on or off (this can be done from the RBA dashboard as well)

Root Factor: you can change the type, scope, threshold, reset Interval, and any other parameters that appear.

Action: you can change the type, scope,duration, and authLevel (as relevant to the chosen root factor). You can base the new rule on the existing templates or create a custom rule. A template-based rule can be edited in the next step.

Account Rules

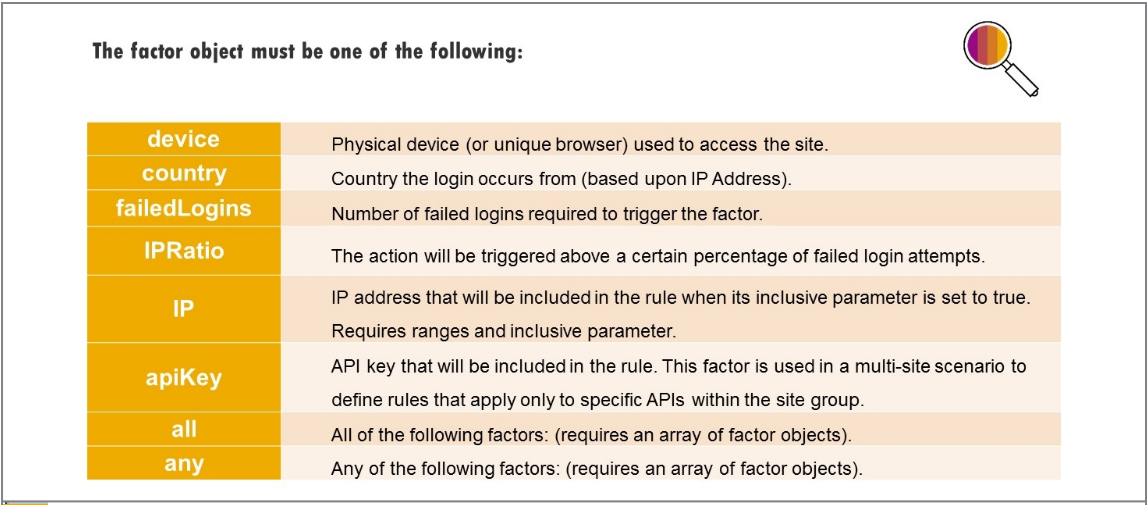
In the Add Rule-Set window, you can edit the following: the unique ID of this rule.

The ID must be unique and not start with an underscore _ .

Status: can be switched on or off (this can be done from the RBA dashboard as well)

Rules: edit the JSON object according to: https://developers.gigya.com/display/GD/Accounts+RBA+Policy+Object

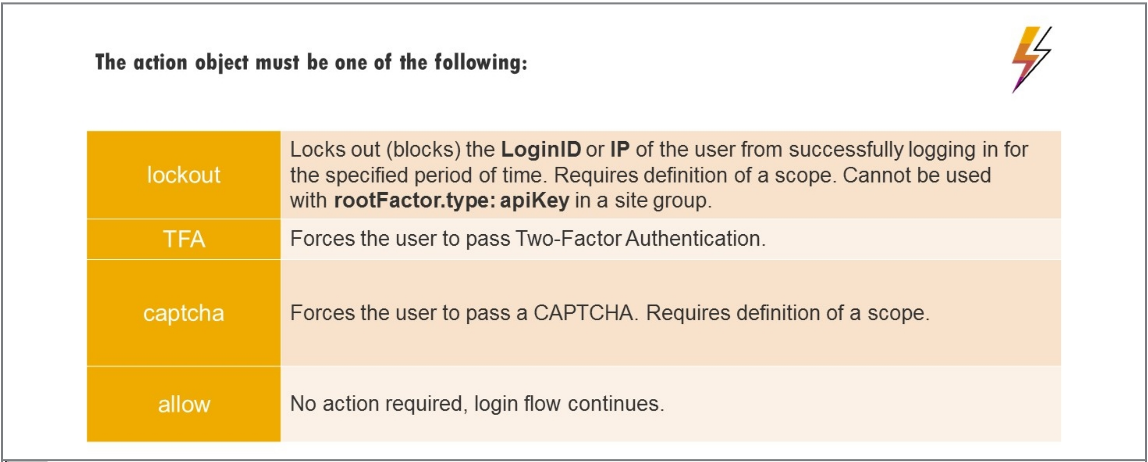
Here you can find some extra policy examples: https://developers.gigya.com/display/GD/Accounts+RBA+Policy+Object#AccountsRBAPolicyObject-PolicyExamples



The factor object must be one of the following:

device	Physical device (or unique browser) used to access the site.
country	Country the login occurs from (based upon IP Address).
failedLogins	Number of failed logins required to trigger the factor.
IPRatio	The action will be triggered above a certain percentage of failed login attempts.
IP	IP address that will be included in the rule when its inclusive parameter is set to true. Requires ranges and inclusive parameter.
apiKey	API key that will be included in the rule. This factor is used in a multi-site scenario to define rules that apply only to specific APIs within the site group.
all	All of the following factors: (requires an array of factor objects).
any	Any of the following factors: (requires an array of factor objects).

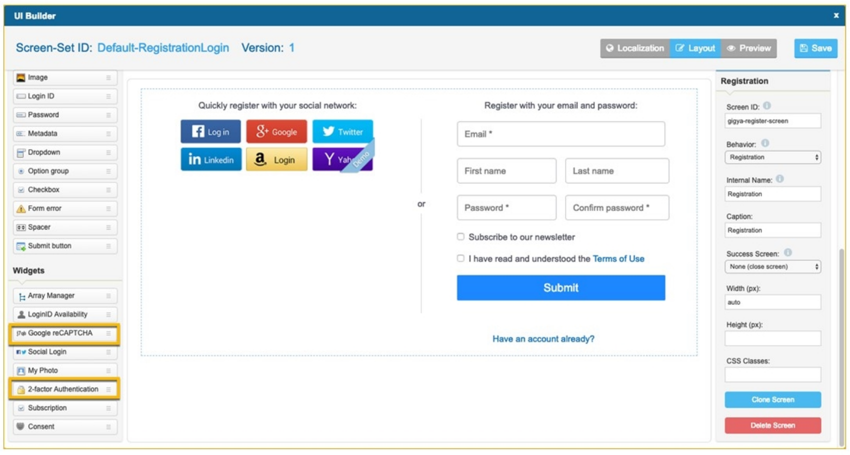
Figure 170: RootFactor Types



The action object must be one of the following:

lockout	Locks out (blocks) the **LoginID** or **IP** of the user from successfully logging in for the specified period of time. Requires definition of a scope. Cannot be used with **rootFactor.type: apiKey** in a site group.
TFA	Forces the user to pass Two-Factor Authentication.
captcha	Forces the user to pass a CAPTCHA. Requires definition of a scope.
allow	No action required, login flow continues.

Figure 171: Action Types



Screen-Set ID: Default-RegistrationLogin Version: 1

Registration

Screen ID: gigya-register-screen

Behavior: Registration

Internal Name: Registration

Caption: Registration

Success Screen: None (close screen)

Width (px): auto

Height (px):

CSS Classes:

Figure 172: Customize Screen-Set to display Captcha and TFA

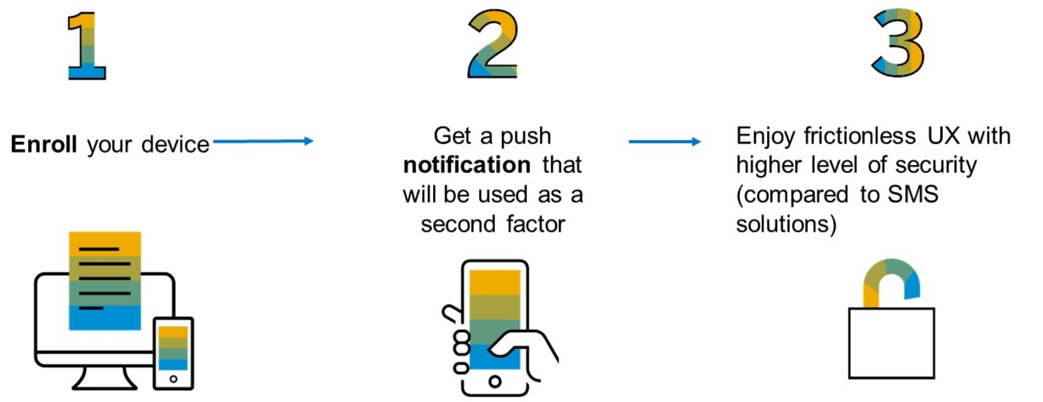
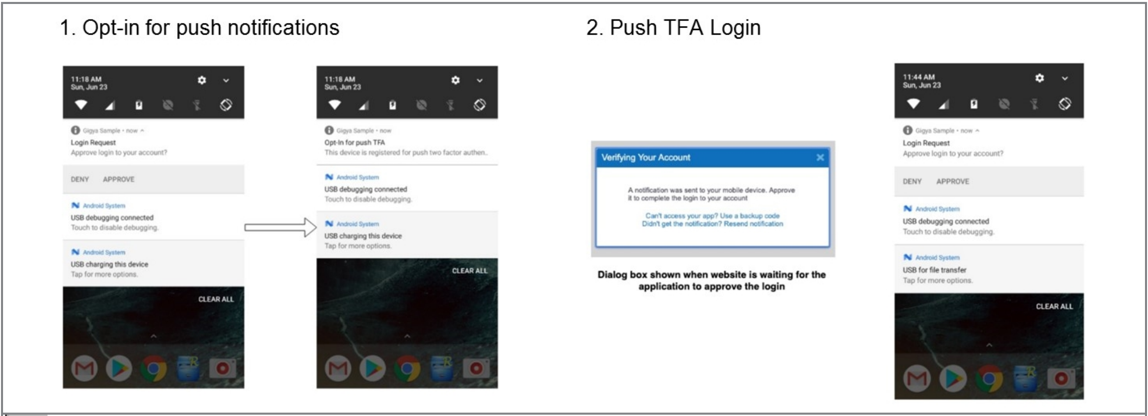

1
Enroll your device
2
Get a push notification that will be used as a second factor
3
Enjoy frictionless UX with higher level of security (compared to SMS solutions)

Figure 173: Push Two-Factor Authentication Flow Overview

- Log in on web with 2nd factor on device
- Stronger security than other current TFA methods (email, SMS, TOTP) since push TFA is paired to a specific physical device and doesn't rely on compromised protocols (SS7) for token delivery
- Gain the added protection of having to authenticate first to the device - guarantees device ownership and proximity
- An experience that end users recognize and are comfortable with
- Push notifications are sent via Google's Firebase Cloud Messaging (FCM) platform, delivering a true cross-platform experience
- Available on the new Mobile SDK

1. Opt-in for push notifications

2. Push TFA Login



Verifying Your Account

A notification was sent to your mobile device. Approve it to complete the login to your account

Can't access your app? Use a backup code
Didn't get the notification? Resend notification

Dialog box shown when website is waiting for the application to approve the login

Figure 174: Push Two-Factor Authentication User Experience

Enabling push notifications requires setting up your app, and configuring the RBA settings in the SAP Customer Data Cloud Console. Push notifications require users to have an active session in your application. A push notification flow can be outlined as follows:

1. A user registers to your site or app and completes their registration as usual (this could require a TFA verification, up to level 20, but doesn't have to).

2. At some point, based on your business logic (e.g. after a user downloads your app and has an active session there, or after a specific app interaction), the user is asked to opt-in to push notifications. This sends a push notification to the mobile application, that the user then needs to approve.

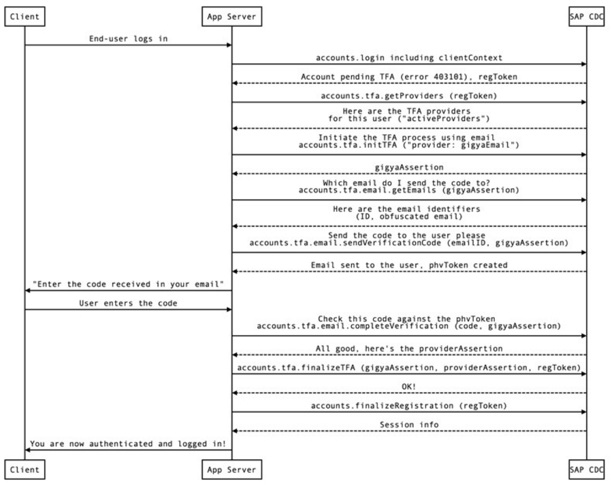
3. After a user opts-in, when later the user tries to login from another device (e.g. desktop), the user receives a push notification to their mobile phone, and can complete login only after approving the notification.

4. Following the push notification opt-in, the regular logic of RBA applies to the user login flows, with push notification used as a second-factor for authentication.

Technical information and configuration: https://developers.gigya.com/display/GD/Risk+Based+Authentication#RiskBasedAuthentication-PushNotification

https://developers.gigya.com/display/GD/Swift+v1.x+TFA+Library#Swiftv1.xTFALibrary-PushTFA

- RBA rules can be applied to **server-side implementations**
- A **clientContext** object needs to be added to the **accounts.login** call
- Once clientContext is passed, RBA applies to the login attempt.



End-user logs in

accounts.login including clientContext

Account pending TFA (error 403101), regToken

accounts.tfa.getProviders (regToken)

Here are the TFA providers for this user ("activeProviders")

Initiate the TFA process using email accounts.tfa.initTFA ("provider: gigyaEmail")

gigyaAssertion

Which email do I send the code to? accounts.tfa.email.getEmails (gigyaAssertion)

Here are the email identifiers (ID, obfuscated email)

Send the code to the user please accounts.tfa.email.sendVerificationCode (emailID, gigyaAssertion)

Email sent to the user, phvToken created

"Enter the code received in your email"

User enters the code

Check this code against the phvToken accounts.tfa.email.completeVerification (code, gigyaAssertion)

All good, here's the providerAssertion

accounts.tfa.finalizeTFA (gigyaAssertion, providerAssertion, regToken)

OK!

accounts.finalizeRegistration (regToken)

Session info

You are now authenticated and logged in!

Figure 175: Server-Side Implementations

Figure 176: Exercise 6

LESSON SUMMARY

You should now be able to:

- Explain the various authentication methods available for the SAP Customer Data Cloud developer

UNIT 7 Customer Consent

UNIT OBJECTIVES

- Manage customer consent

Understanding Customer Consent

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Manage customer consent

Regulations

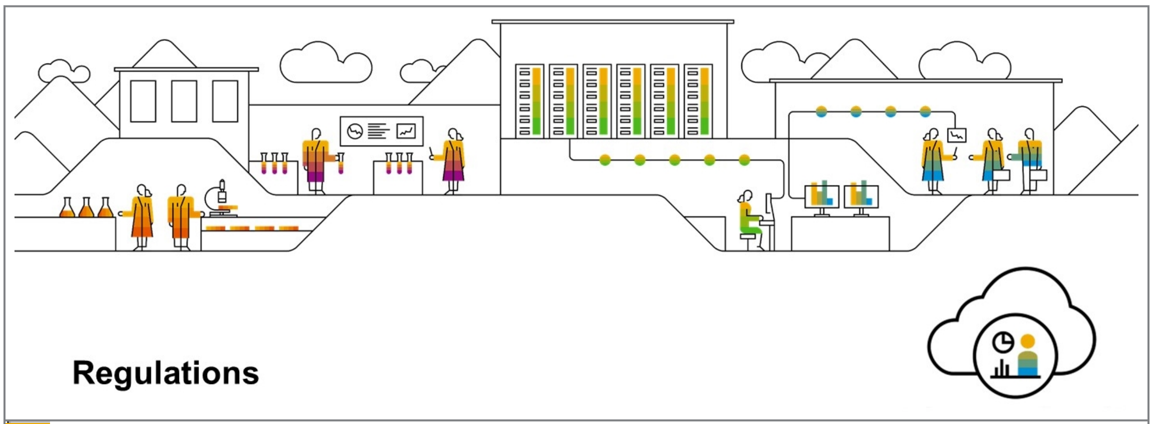
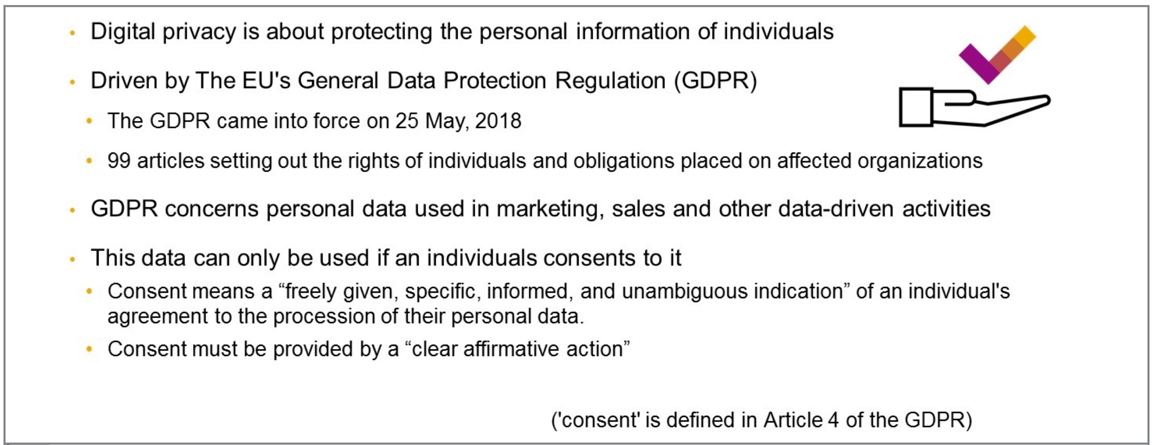


Figure 177: Regulations

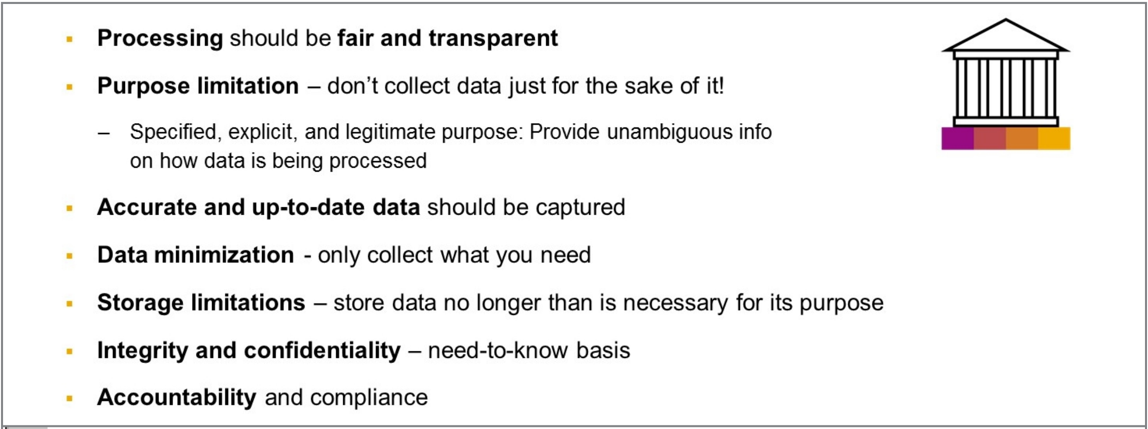


- Digital privacy is about protecting the personal information of individuals
- Driven by The EU's General Data Protection Regulation (GDPR)
  - The GDPR came into force on 25 May, 2018
  - 99 articles setting out the rights of individuals and obligations placed on affected organizations
- GDPR concerns personal data used in marketing, sales and other data-driven activities
- This data can only be used if an individuals consents to it
  - Consent means a “freely given, specific, informed, and unambiguous indication” of an individual's agreement to the procession of their personal data.
  - Consent must be provided by a “clear affirmative action”

('consent' is defined in Article 4 of the GDPR)

Figure 178: Customer Consent

The quoted parts of the definition of consent come from Article 4 of the GDPR. The full text of this definition is:'consent' of the data subject means any freely given, specific, informed and unambiguous indication of the data subject's wishes by which he or she, by a statement or by a clear affirmative action, signifies agreement to the processing of personal data relating to him or her;



- **Processing** should be **fair and transparent**
- **Purpose limitation** – don't collect data just for the sake of it!
  - Specified, explicit, and legitimate purpose: Provide unambiguous info on how data is being processed
- **Accurate and up-to-date data** should be captured
- **Data minimization** - only collect what you need
- **Storage limitations** – store data no longer than is necessary for its purpose
- **Integrity and confidentiality** – need-to-know basis
- **Accountability** and compliance

Figure 179: GDPR Principles

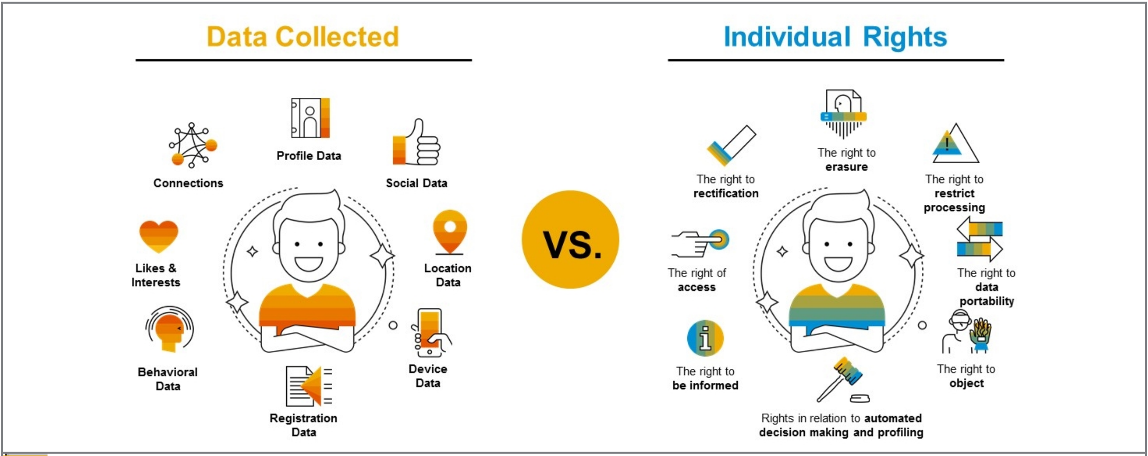
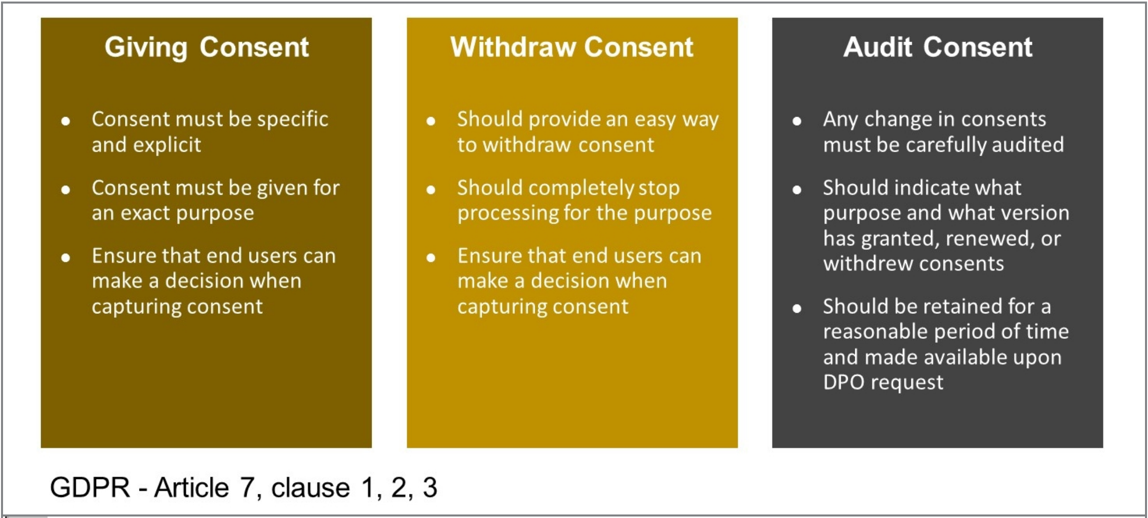


Figure 180: GDPR is about putting the customer in control of their data



Giving Consent

- Consent must be specific and explicit
- Consent must be given for an exact purpose
- Ensure that end users can make a decision when capturing consent

Withdraw Consent

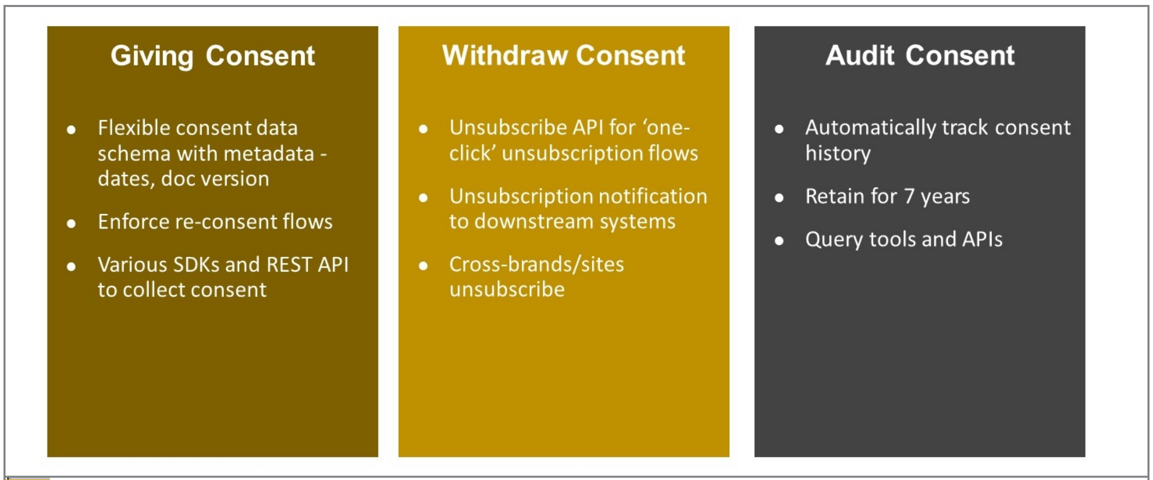
- Should provide an easy way to withdraw consent
- Should completely stop processing for the purpose
- Ensure that end users can make a decision when capturing consent

Audit Consent

- Any change in consents must be carefully audited
- Should indicate what purpose and what version has granted, renewed, or withdrew consents
- Should be retained for a reasonable period of time and made available upon DPO request

GDPR - Article 7, clause 1, 2, 3

Figure 181: Consent - Recap



Giving Consent

- Flexible consent data schema with metadata - dates, doc version
- Enforce re-consent flows
- Various SDKs and REST API to collect consent

Withdraw Consent

- Unsubscribe API for 'one-click' unsubscription flows
- Unsubscription notification to downstream systems
- Cross-brands/sites unsubscribe

Audit Consent

- Automatically track consent history
- Retain for 7 years
- Query tools and APIs

Figure 182: SAP Customer Consent and Profile Platform

https://cx.sap.com/en/products/customer-data-cloud/consent

SAP Customer Consent

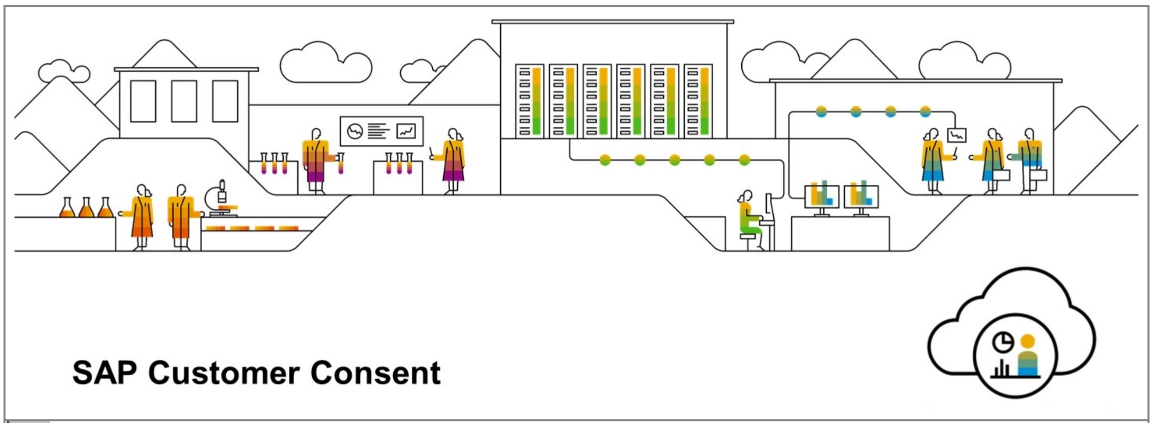
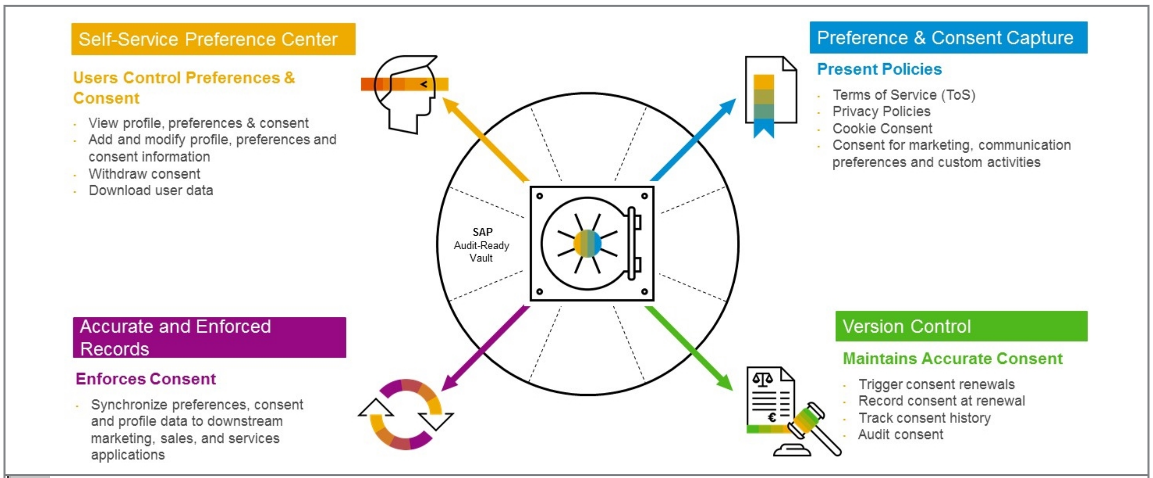


Figure 183: SAP Customer Consent



Self-Service Preference Center

Users Control Preferences & Consent

- View profile, preferences & consent
- Add and modify profile, preferences and consent information
- Withdraw consent
- Download user data

Preference & Consent Capture

Present Policies

- Terms of Service (ToS)
- Privacy Policies
- Cookie Consent
- Consent for marketing, communication preferences and custom activities

SAP Audit-Ready Vault

Accurate and Enforced Records

Enforces Consent

- Synchronize preferences, consent and profile data to downstream marketing, sales, and services applications

Version Control

Maintains Accurate Consent

- Trigger consent renewals
- Record consent at renewal
- Track consent history
- Audit consent

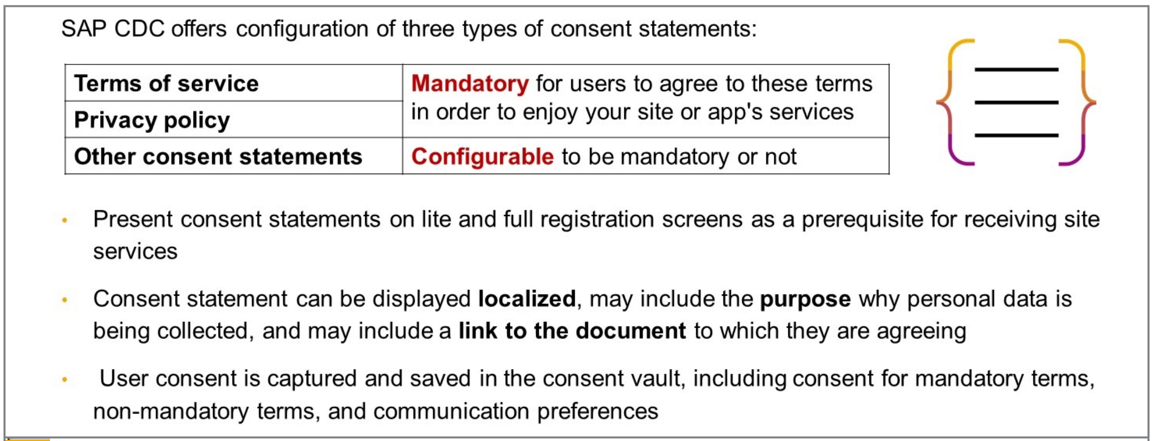
Figure 184: SAP Customer Consent Overview

Benefits:

- Supports compliance with data privacy laws: require consent to terms of service and privacy policies as a prerequisite for using some of your site services
- Additional optional consent statements allow flexibility in your relationship with the customer
- Communication preferences allow users to subscribe to various communication channels with the option of requiring double opt-in as proof of the user's intention to subscribe
- Displays clearly to users which data was saved to their account and provides the option to withdraw consent and manage communication preferences
- Consent Vault contains an audit of the customer's agreement to site policies and the version to which they consented
- Unified user database enables presenting a holistic view of the customer, including data gathered from various platforms
- Synchronize consent-based user data to 3rd-party platforms using IdentitySync or GConnectors

Consent Management

Figure 185: Consent Management

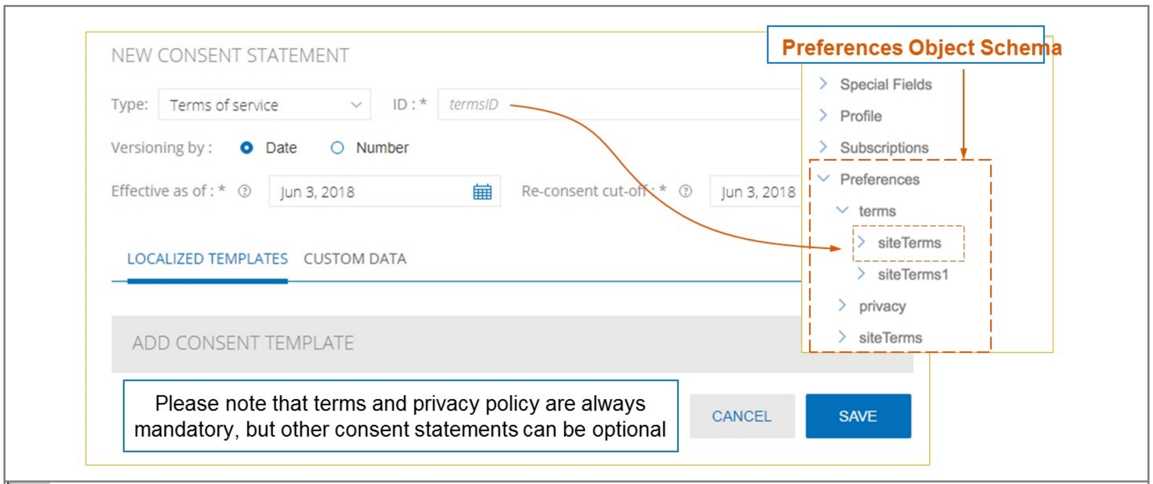


SAP CDC offers configuration of three types of consent statements:

Terms of service	**Mandatory** for users to agree to these terms in order to enjoy your site or app's services
Privacy policy	
Other consent statements	**Configurable** to be mandatory or not

- Present consent statements on lite and full registration screens as a prerequisite for receiving site services
- Consent statement can be displayed **localized**, may include the **purpose** why personal data is being collected, and may include a **link to the document** to which they are agreeing
- User consent is captured and saved in the consent vault, including consent for mandatory terms, non-mandatory terms, and communication preferences

Figure 186: Consent Statements


NEW CONSENT STATEMENT

Type: Terms of service   ID : * termsID

Versioning by : [✓ Date] Number

Effective as of : * Jun 3, 2018   Re-consent cut-off : * Jun 3, 2018

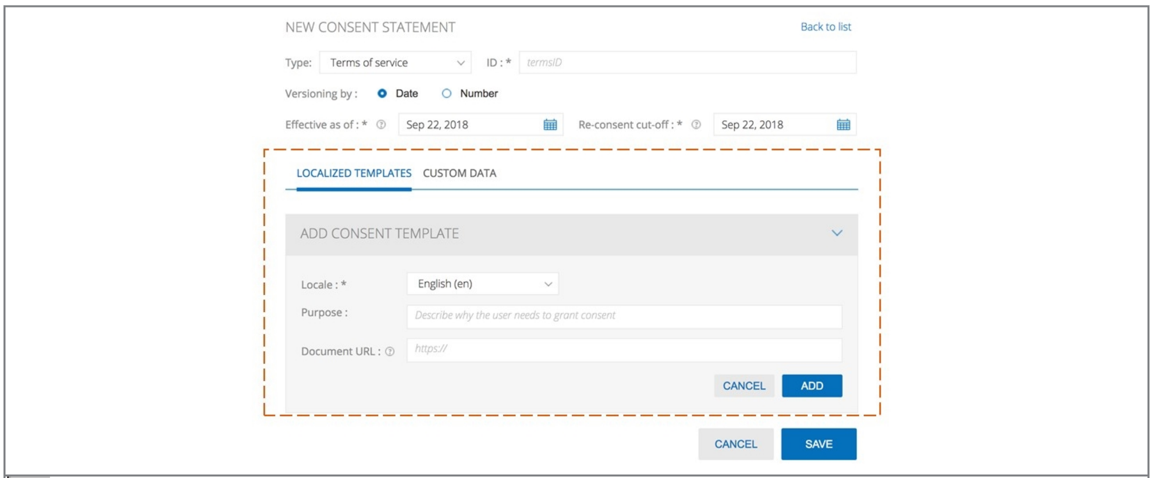
Please note that terms and privacy policy are always mandatory, but other consent statements can be optional

Figure 187: Creating a new consent statement

Registration as a Service -> Consent -> Add

1. Select Type of the consent (Terms of Service, Privacy Policy, or Others)
2. In the ID field, enter a unique identifier for the term
3. If you create a new term, the new term will reflect in Preferences Object Schema under terms, as shown in the screenshot above
4. If you create a new privacy policy, it will be under privacy.

Note: after adding a consent statement it cannot be removed. It can only be deactivated.


NEW CONSENT STATEMENT

Type: Terms of service   ID : * termsID

Versioning by : [✓ Date] Number

Effective as of : * Sep 22, 2018   Re-consent cut-off : * Sep 22, 2018

Locale : * English (en)

Purpose : Describe why the user needs to grant consent

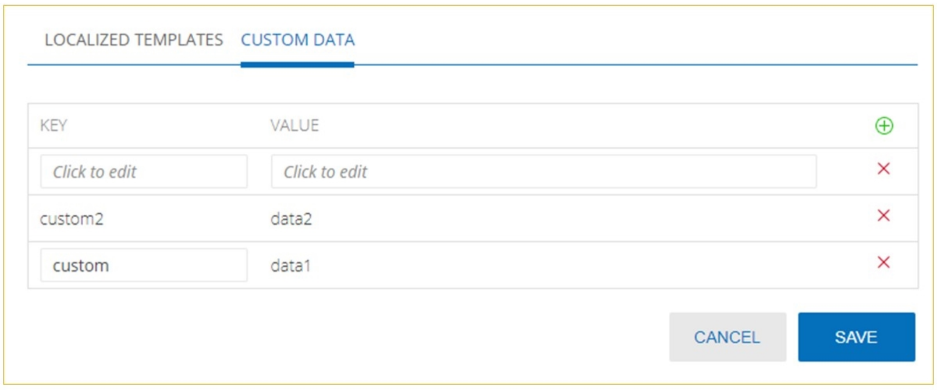
Document URL : https://

Figure 188: Creating a Consent Statement - Localized Templates

You can add a localized template that includes a locale, the purpose of this statement, and a link to the document to which the user is agreeing.

Any document URLs provided must be persistent and the document must be available on the defined URI for as long as required by the country of residence of the end-user. It is the client's responsibility to maintain accurate records of any consent templates that were agreed to by the end-user. The document must be a PDF file.

The legal statement and purpose will be displayed for the relevant end-user in the Consent History tab of Identity Access and in the Consent Vault, but will not be visible on the user account via accounts.getAccountInfo.



KEY	VALUE
Click to edit	Click to edit
custom2	data2
custom	data1

Figure 189: Creating a Consent Statement - Custom Data

The custom data will be available on the account (when calling accounts.search or accounts.getAccountInfo) and will be audited in the consent vault. These pairs will be saved to the Preferences Schema Object.

The maximum number of characters for the key is 20 and 256 for the value. The maximum number of custom key-value pairs per consent statement is 50.

- Using Customer Data Cloud's screens and UI Builder to maintain the consent statements
- Any consent statement that is set to active has to be included in your registration and registration completion screens

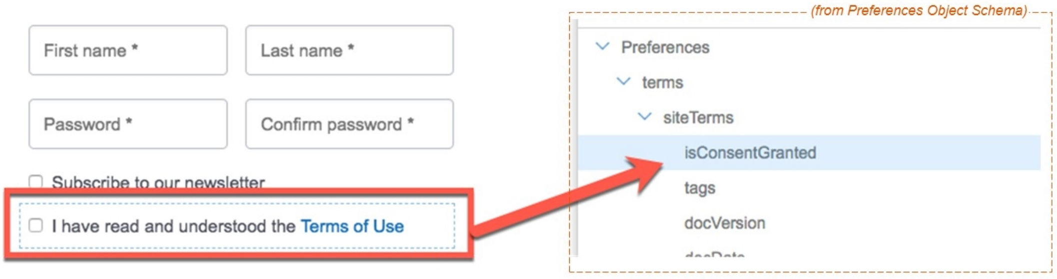
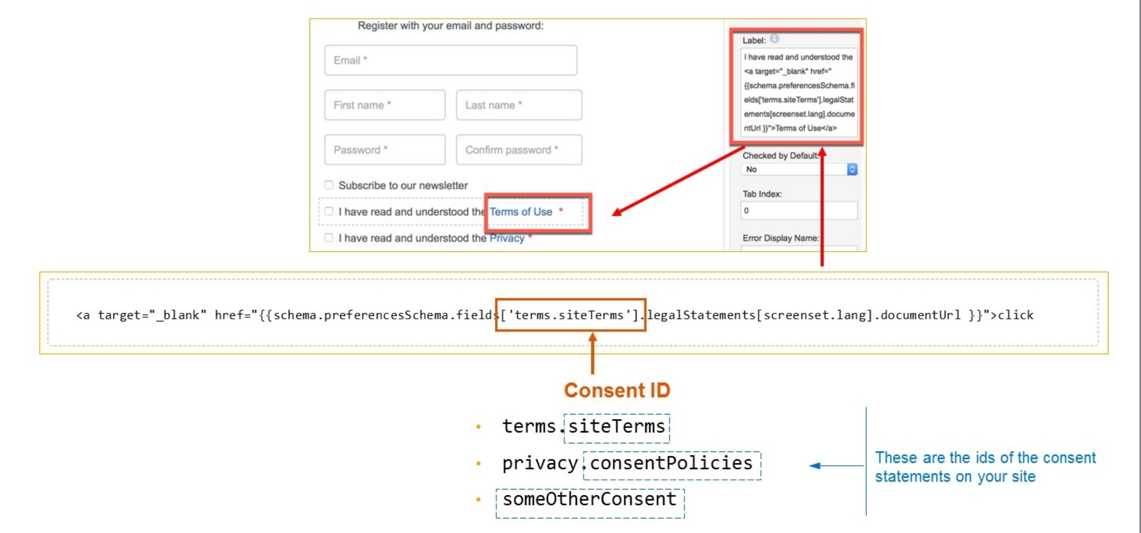


Figure 190: Add the Statement to a Screen

Note that any consent statement that is set to "active" for your site, including a non-mandatory consent statement, has to be included in your registration and registration completion screens. Otherwise, users will not be able to complete their registration.


Register with your email and password:

Email *

First name * | Last name *

Password * | Confirm password *

☐ Subscribe to our newsletter

☐ I have read and understood the Terms of Use *

☐ I have read and understood the Privacy *

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

Checked by Default: No

Tab Index: 0

Error Display Name:

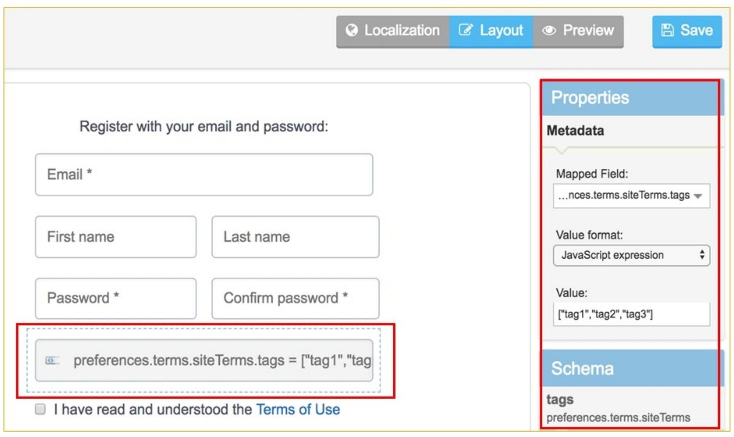
```
<a target="_blank" href="{{schema.preferencesSchema.fields['terms.siteTerms'].legalStatements[screenset.lang].documentUrl }}">click
```

- terms.siteTerms
- privacy.consentPolicies
- someOtherConsent

Figure 191: Update the consent document link on the registration form

When a user clicks on the Terms of Use link, the consent document linked to the consent statement should open.

- Use the ‘tags’ property
- Tags can be mapped to a metadata component. They also support multiple data values using an array structure
- Tag values are visible in both the consent vault and Identity Access, but not on the user account



Register with your email and password:

Email *

First name | Last name

Password * | Confirm password *

preferences.terms.siteTerms.tags = ["tag1","tag

☐ I have read and understood the Terms of Use

Properties

Metadata

Mapped Field: ...nces.terms.siteTerms.tags

Value format: JavaScript expression

Value: ["tag1","tag2","tag3"]

Schema

tags
preferences.terms.siteTerms

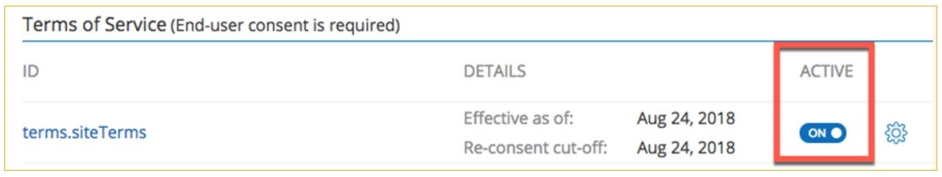
Figure 192: Add Dynamic Data to the Consent Record in the Consent Vault

If you want to set additional dynamic data against consent records in the consent vault, you can do so by using the 'tags' property.

Tags can be mapped to a metadata component and can support multiple data values using an array structure.

Tag values are only visible in the consent vault and not against the user account.

One example use case of tags is that with tags we can define different values for different screen sets. This way, once a user updates consent data, we know which exact screen exactly this consent was initiated from and can capture this info in the consent record.

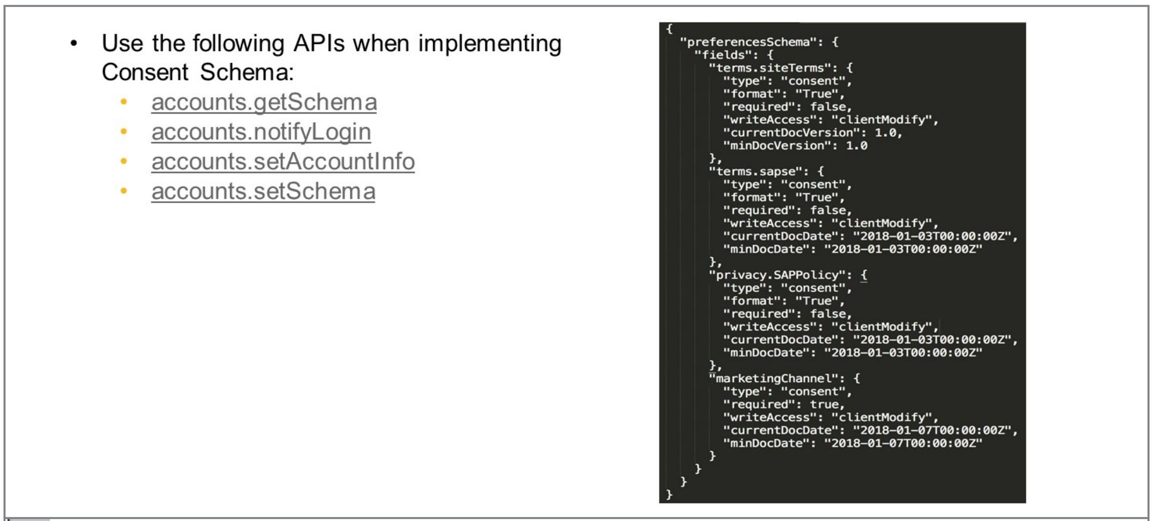


Terms of Service (End-user consent is required)

ID	DETAILS		ACTIVE
terms.siteTerms	Effective as of:	Aug 24, 2018	ON
	Re-consent cut-off:	Aug 24, 2018	

Figure 193: Activating a Consent Statement

Activate the consent statement and make the field required in the UI Builder



- Use the following APIs when implementing Consent Schema:
  - accounts.getSchema
  - accounts.notifyLogin
  - accounts.setAccountInfo
  - accounts.setSchema

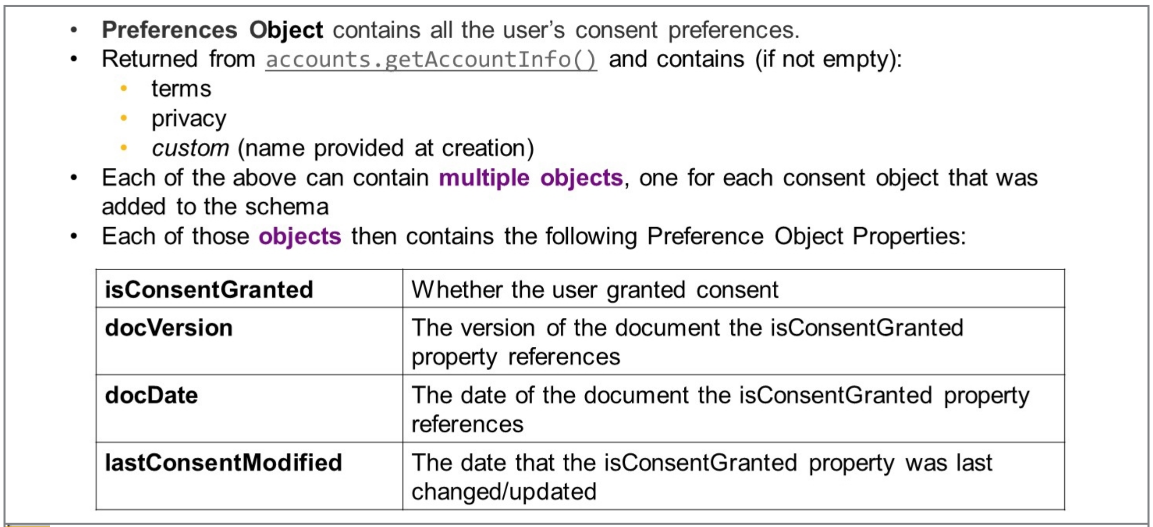
```
{
  "preferencesSchema": {
    "fields": {
      "terms.siteTerms": {
        "type": "consent",
        "format": "True",
        "required": false,
        "writeAccess": "clientModify",
        "currentDocVersion": 1.0,
        "minDocVersion": 1.0
      },
      "terms.sapse": {
        "type": "consent",
        "format": "True",
        "required": false,
        "writeAccess": "clientModify",
        "currentDocDate": "2018-01-03T00:00:00Z",
        "minDocDate": "2018-01-03T00:00:00Z"
      },
      "privacy.SAPPolicy": {
        "type": "consent",
        "format": "True",
        "required": false,
        "writeAccess": "clientModify",
        "currentDocDate": "2018-01-03T00:00:00Z",
        "minDocDate": "2018-01-03T00:00:00Z"
      },
      "marketingChannel": {
        "type": "consent",
        "required": true,
        "writeAccess": "clientModify",
        "currentDocDate": "2018-01-07T00:00:00Z",
        "minDocDate": "2018-01-07T00:00:00Z"
      }
    }
  }
}
```

Figure 194: Consent Schema overview

This is a extract from the accounts.getSchema() JSON response and the aim of the slide is to show the preferences schema.

In the response you will see if some consents are already defined for your consent application. The tags will also be shown on the agenda if you have defined them to be added on the consent records. See the previous slide as an example.

In this example, some fields, like terms.siteTerm, are marked as required and some are not. You can see each field has a type, format, required, and currentDocDate.



- **Preferences Object** contains all the user's consent preferences.
- Returned from `accounts.getAccountInfo()` and contains (if not empty):
  - terms
  - privacy
  - *custom* (name provided at creation)
- Each of the above can contain **multiple objects**, one for each consent object that was added to the schema
- Each of those **objects** then contains the following Preference Object Properties:

isConsentGranted	Whether the user granted consent
docVersion	The version of the document the isConsentGranted property references
docDate	The date of the document the isConsentGranted property references
lastConsentModified	The date that the isConsentGranted property was last changed/updated

Figure 195: Preferences Object

Version Control

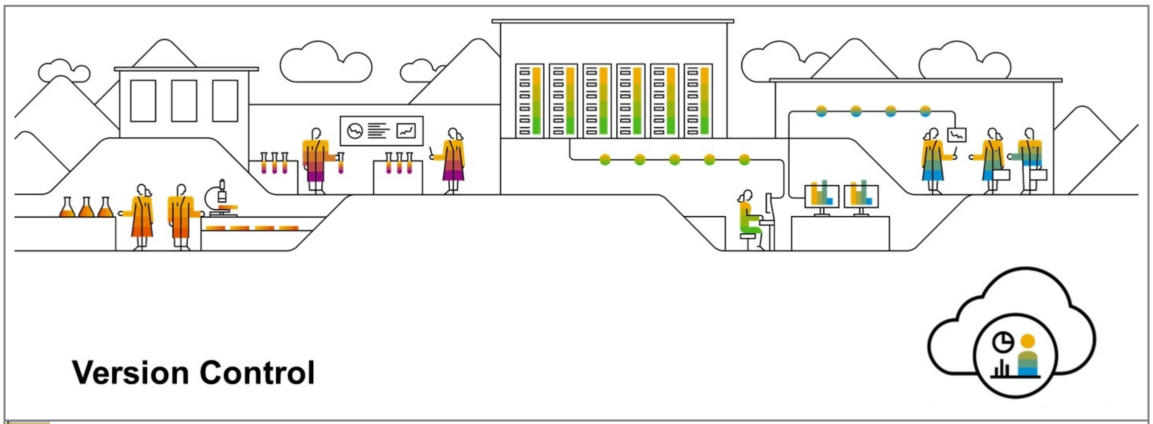
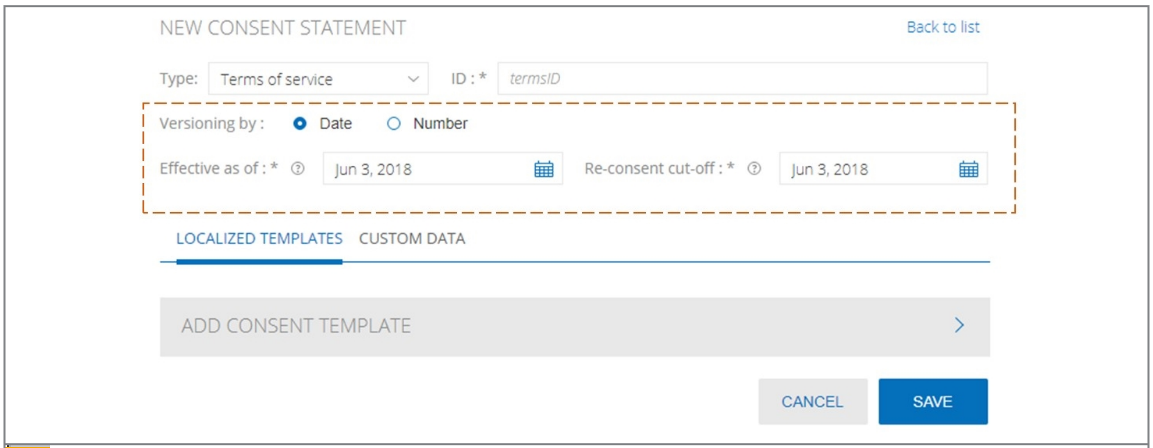


Figure 196: Version Control



NEW CONSENT STATEMENT

Back to list

Type: Terms of service

ID : * termsID

Versioning by : [✓ Date] Number

Effective as of : * Jun 3, 2018

Re-consent cut-off : * Jun 3, 2018

LOCALIZED TEMPLATES CUSTOM DATA

ADD CONSENT TEMPLATE

CANCEL SAVE

Figure 197: Creating a New Consent Statement - Versioning by

You can choose versioning by date or number.

Once you save the new statement, you cannot change this definition.

A re-consent cut-off should be added with any major consent change that requires existing users to re-consent

Validating the User's Consent Status

Consent management includes a built-in mechanism for ensuring that users who login to your site have a valid consent in place for the relevant mandatory statements:

- A new user registers to your site but does not agree to the terms of service. The user will not be able to submit the registration screen
- A user registered to your site before the consent module was implemented and has no valid consent statements associated with their account. The next time they log in, the "Registration Completion" screen will be displayed, requesting the relevant agreements.
- A user has agreed to your site consent statements and is still logged in, but the site admins have changed the version of one of the consent statements. During their logged-in session, the "Registration Completion" screen will be displayed, requesting their re-consent.

To ensure these scenarios take place, you must make sure to include the relevant consent statements on all registration and registration completion screens and set a value to the verifyLoginInterval parameter in the WebSDK Configuration.

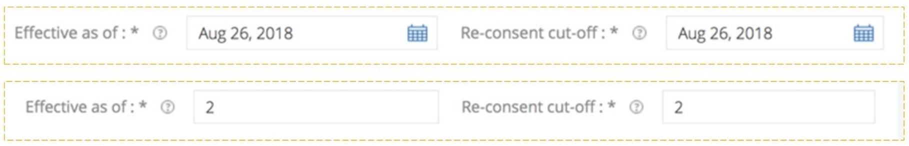
Making a **minor** change to a statement

- Update the field: `Effective as of`

Making a **major** change to a statement

- Update the field: `Effective as of`
- Update the field: `Re-consent cut-off`

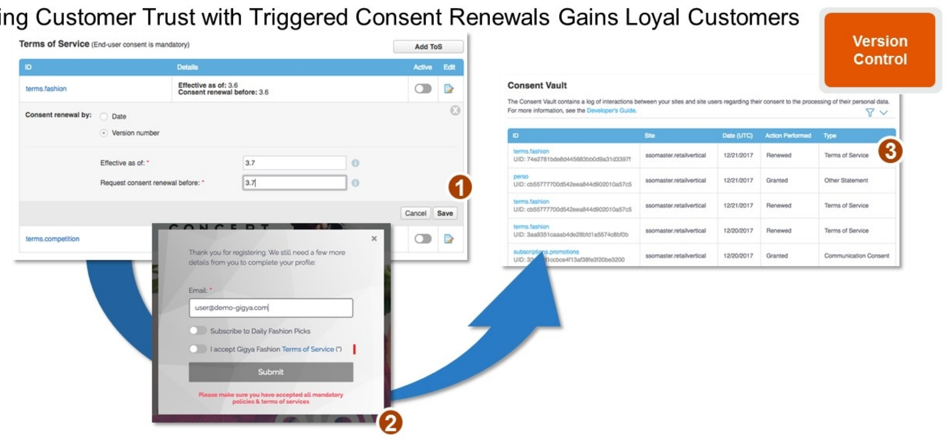
Users who consented to a version that is earlier than the number or date specified under `Re-consent cut-off`, will be required to re-consent



Effective as of : *	Aug 26, 2018	Re-consent cut-off : *	Aug 26, 2018
Effective as of : *	2	Re-consent cut-off : *	2

Figure 198: Major vs. Minor Changes

Earning Customer Trust with Triggered Consent Renewals Gains Loyal Customers



Terms of Service (End-user consent is mandatory)

ID	Details	Active	Edit
terms.fashion	Effective as of: 3.6 Consent renewal before: 3.6		
terms.competition			

Consent renewal by: Date [✓ Version number]

Effective as of: * 3.7

Request consent renewal before: * 3.7

Thank you for registering. We still need a few more details from you to complete your profile:

Email: * user@demo-gigya.com

Subscribe to Daily Fashion Picks

I accept Gigya Fashion Terms of Service (*)

Please make sure you have accepted all mandatory policies & terms of services

Consent Vault

The Consent Vault contains a log of interactions between your sites and site users regarding their consent to the processing of their personal data. For more information, see the Developer's Guide.

ID	Site	Date (UTC)	Action Performed	Type
terms.fashion UID: 74e2781bde8d445683bb0d9a31d3397f	ssomaster.retailvertical	12/21/2017	Renewed	Terms of Service
perso UID: cb55777700d542eea844d902010a57c5	ssomaster.retailvertical	12/21/2017	Granted	Other Statement
terms.fashion UID: cb55777700d542eea844d902010a57c5	ssomaster.retailvertical	12/21/2017	Renewed	Terms of Service
terms.fashion UID: 3aa9351caaab4de28bfd1a5574c8bf0b	ssomaster.retailvertical	12/20/2017	Renewed	Terms of Service
subscriptions.promotions UID: 33[illegible]d1ccbce4f13af38fe3f20be3200	ssomaster.retailvertical	12/20/2017	Granted	Communication Consent

Figure 199: SAP Customer Consent - Version Control

The consent management tool gives you the option to update the version of the consent statement that is currently in effect, and the required date for consent renewal.

For minor changes: simply, create a new version by updating the Effective as of field in the consent management dashboard

For a new version of consent: that require existing users to re-consent, update both the Effective as of field (the active version number or date), and the Re-consent cut-off field.

Consent Vault

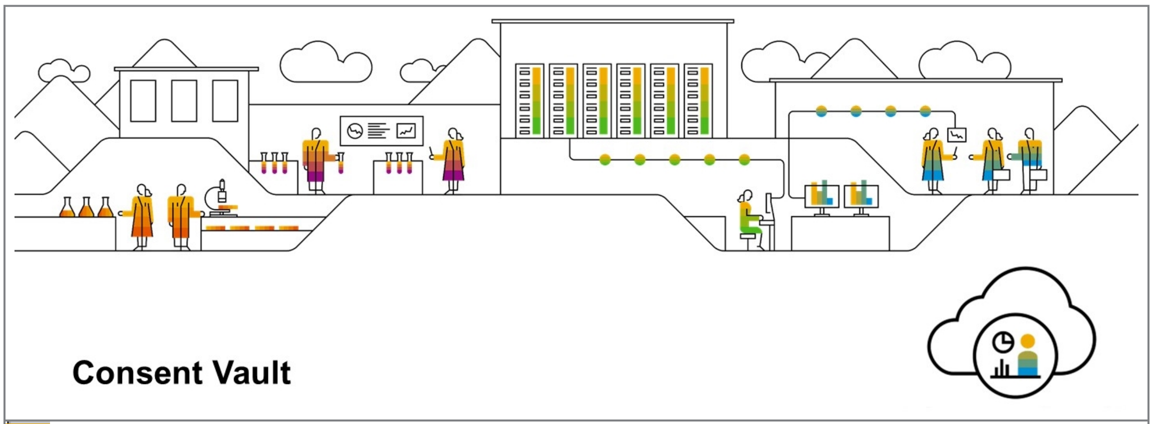
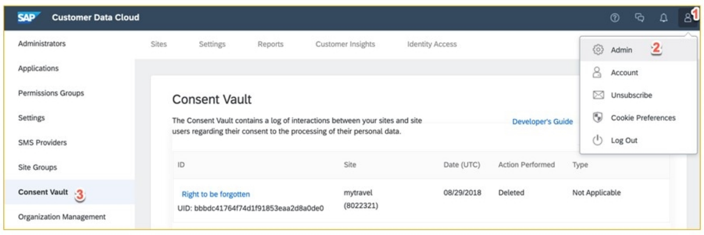


Figure 200: Consent Vault

- A log of interactions between your site and site users regarding their consent to the processing of their personal data
- View and search the history of all consent objects on your site
- See the status of a user's consent
- Captures user actions related to consent of the following types of documents and interactions
  - Terms of service
  - Privacy policy
  - Other consent statements
  - Communication preferences
    - e.g. opting in or out of a newsletter
  - Right to be forgotten



Consent Vault

The Consent Vault contains a log of interactions between your sites and site users regarding their consent to the processing of their personal data.

ID	Site	Date (UTC)	Action Performed	Type
Right to be forgotten UID: bbbdc41764f74d1f91853eaa2d8a0de0	mytravel (8022321)	08/29/2018	Deleted	Not Applicable

Figure 201: Consent Vault

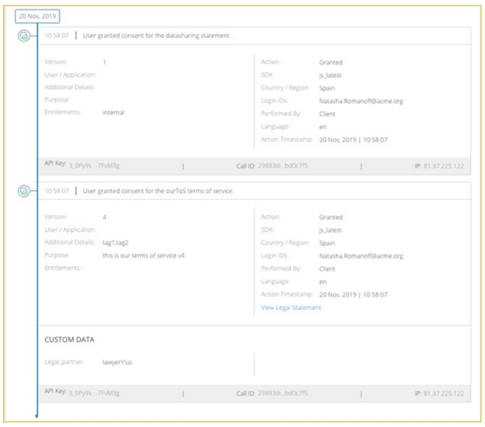
When users re-consent, the action is captured and saved to the Consent Vault as a "Renewed" consent.

New users consent to the active version, and that is saved to the Consent Vault as a "Granted" consent.

You can use the vault to easily view and search the history of all consent interactions on your site

For each interaction, there is full traceability of the interaction, including

- What **text** the customer saw at the time
- Which **version** the customer accepted
- A **timestamp** of when the customer accepted it
- Which **IP** address the customer accepted it from
- The login **ID** that performed the action


20 Nov, 2019

10:58:07 | User granted consent for the datasharing statement.

Version:	1	Action:	Granted
User / Application:		SDK:	js_latest
Additional Details:		Country / Region:	Spain
Purpose:		Login IDs:	Natasha.Romanoff@acme.org
Entitlements:	internal	Performed By:	Client
		Language:	en
		Action Timestamp:	20 Nov, 2019 | 10:58:07

API Key: 3_0Py9v...-7FvM3g | Call ID: 29883d...bd0c7f5 | IP: 81.37.225.122

10:58:07 | User granted consent for the ourToS terms of service.

Version:	4	Action:	Granted
User / Application:		SDK:	js_latest
Additional Details:	tag1,tag2	Country / Region:	Spain
Purpose:	this is our terms of service v4	Login IDs:	Natasha.Romanoff@acme.org
Entitlements:		Performed By:	Client
		Language:	en
		Action Timestamp:	20 Nov, 2019 | 10:58:07
		View Legal Statement	

CUSTOM DATA

Legal_partner: lawyer'r'us

API Key: 3_0Py9v...-7FvM3g | Call ID: 29883d...bd0c7f5 | IP: 81.37.225.122

Figure 202: What is recorded against each consent record in the Consent Vault?

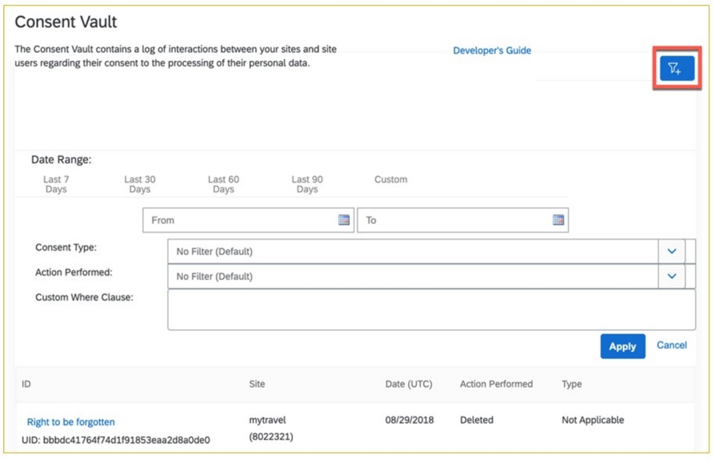
Several options for flexibly customizing the consent record to your needs:

- Custom data: Added to the schema of the consent object. Allows you to add any additional information to the object, such as the legal entity that maintains it, the admin who created it, or for adding a common value to consent records
- Tags: Appended to a specific consent interaction. For example, if the same consent statement is added to two different screens (registration completion and profile update), the tags can indicate the screen on which the user granted their consent, i.e., to which interaction the consent corresponds
- Entitlements: An addition or appendix to the consent interaction. An easy way to ask for additional input from a user (additional to consent they have already given), without having to trigger the re-consent process. Users can only grant entitlements if they have agreed to the original consent statement

https://developers.gigya.com/display/GD/Consent+Management#ConsentManagement-CustomizingConsent-RelatedData

Filter by

- Date Range
- Consent Type
- Action Performed
- Custom Where Clause


Consent Vault

The Consent Vault contains a log of interactions between your sites and site users regarding their consent to the processing of their personal data.

Developer's Guide

Date Range:

Last 7 Days | Last 30 Days | Last 60 Days | Last 90 Days | Custom

From | To

Consent Type: No Filter (Default)

Action Performed: No Filter (Default)

Custom Where Clause:

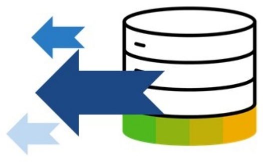
Apply | Cancel

ID	Site	Date (UTC)	Action Performed	Type
Right to be forgotten UID: bbbdc41764f74d1f91853eaa2d8a0de0	mytravel (8022321)	08/29/2018	Deleted	Not Applicable

Figure 203: Consent Vault Filters

query=SELECT * FROM auditLogEvent

```
{
 "callId": "fa6f617403df4b3fa00edfad956773eb",
 "errorCode": 0,
 "apiVersion": 2,
 "statusCode": 200,
  ...
 "results": [ ...
   {
        "legalStatementURL": "https://www.sap.com/corporate/en/legal/terms-of-use.pdf",
        "callID": "aa4b1ee5d15140c1880b92c80d00920d",
        "purpose": "tos version 1",
        "originatingSiteID": "8197434",
        "@type": "auditLogEvent",
        "customData": [],
        "path": "terms.ourtos",
        "uid": "e681feacc2a9445cac7474d6eb7f3297",
        ...
        "originatingApiKey": "3_0e5CZIMK2OlbuL646.......",
        "@datacenter": "eu1",
        "loginIDs": "newreg@sap.com"
   }
 ...
]}
```

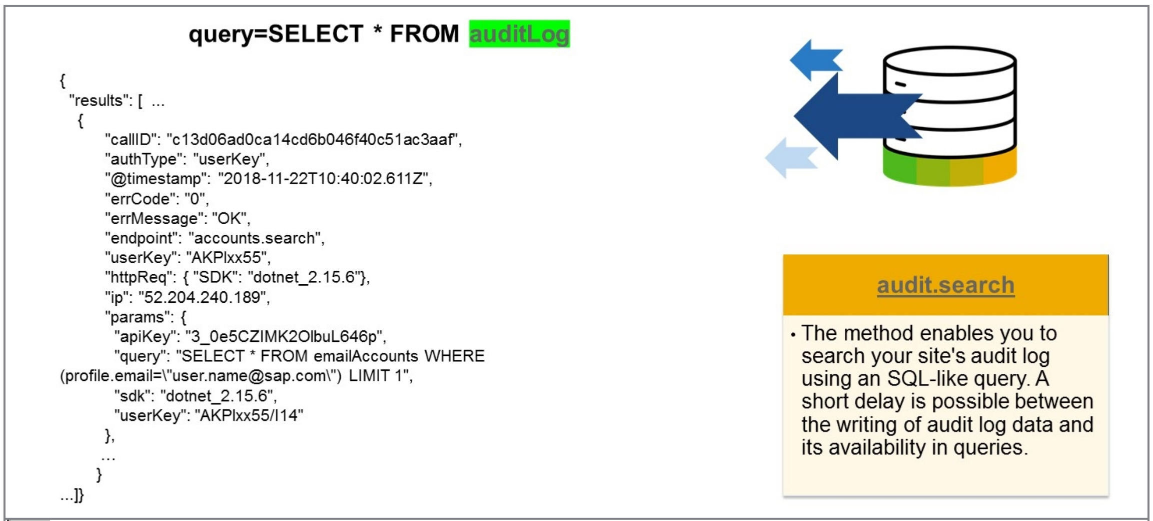


audit.search

- The method enables you to search your site's audit log using an SQL-like query. A short delay is possible between the writing of audit log data and its availability in queries.

Figure 204: Querying the Consent Vault using the REST API

https://developers.gigya.com/display/GD/audit.search



query=SELECT * FROM auditLog

```
{
 "results": [  ...
  {
      "callID": "c13d06ad0ca14cd6b046f40c51ac3aaf",
      "authType": "userKey",
      "@timestamp": "2018-11-22T10:40:02.611Z",
      "errCode": "0",
      "errMessage": "OK",
      "endpoint": "accounts.search",
      "userKey": "AKPlxx55",
      "httpReq": { "SDK": "dotnet_2.15.6"},
      "ip": "52.204.240.189",
      "params": {
        "apiKey": "3_0e5CZIMK2OlbuL646p",
        "query": "SELECT * FROM emailAccounts WHERE
(profile.email=\"user.name@sap.com\") LIMIT 1",
        "sdk": "dotnet_2.15.6",
        "userKey": "AKPlxx55/I14"
      },
      ...
    }
...]}
```

audit.search

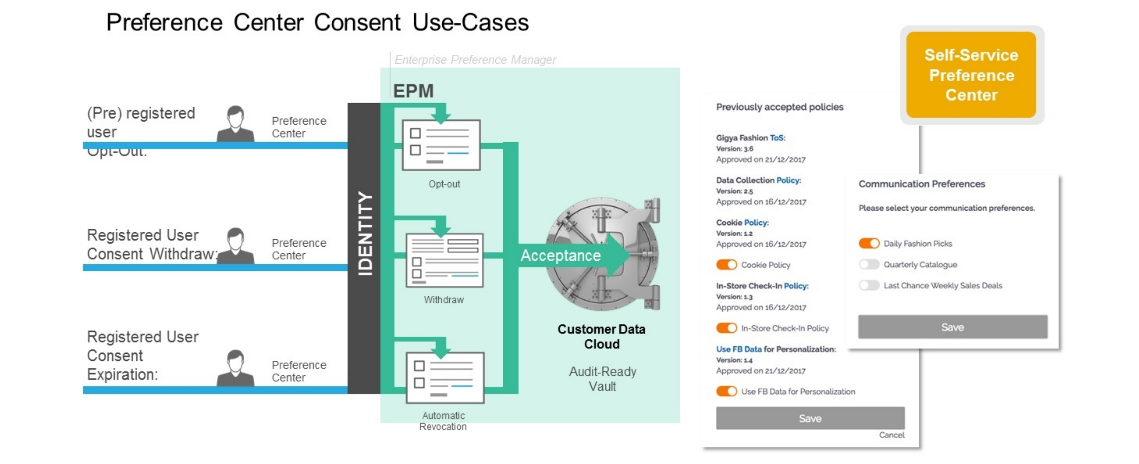
- The method enables you to search your site's audit log using an SQL-like query. A short delay is possible between the writing of audit log data and its availability in queries.

Figure 205: Querying the Auditlog using the REST API

https://developers.gigya.com/display/GD/audit.search

Self-Service Preference Center

Figure 206: Self-Service Preference Center



Previously accepted policies

Gigya Fashion ToS:
Version: 3.6
Approved on 21/12/2017

Data Collection Policy:
Version: 2.5
Approved on 16/12/2017

Cookie Policy:
Version: 1.2
Approved on 16/12/2017

Cookie Policy

In-Store Check-In Policy:
Version: 1.3
Approved on 16/12/2017

In-Store Check-In Policy

Use FB Data for Personalization:
Version: 1.4
Approved on 21/12/2017

Use FB Data for Personalization

Save
Cancel

Communication Preferences

Please select your communication preferences.

Daily Fashion Picks
Quarterly Catalogue
Last Chance Weekly Sales Deals

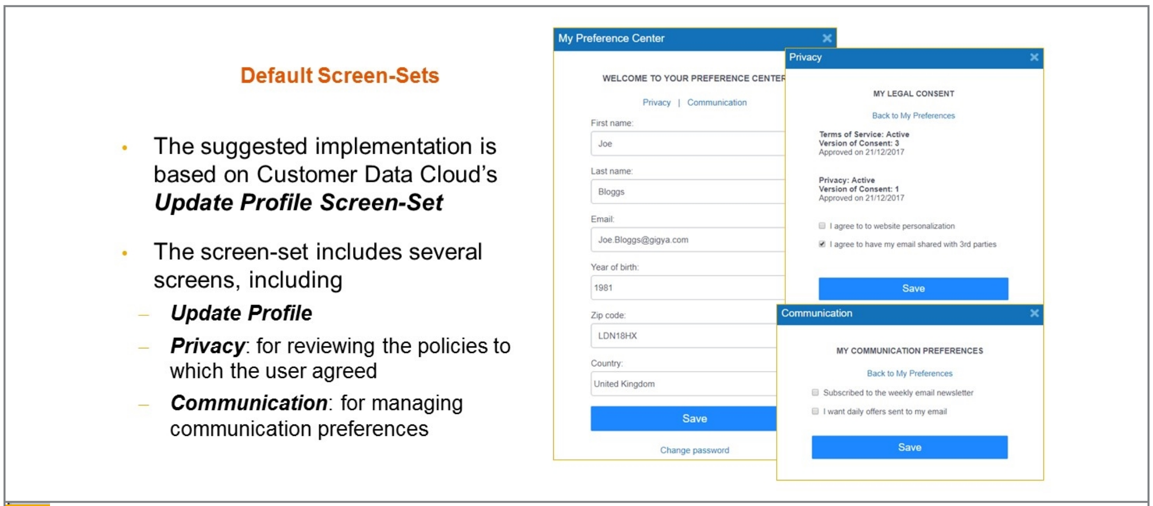
Save

Figure 207: SAP Customer Consent - Self-Service Preference Center

The preference center is your main point of contact with your customer.

Typical use cases :

- Ensuring that new users are consenting to Terms of Service, Privacy Policy, marketing communications, and data processors when signing up for an account
- Ensuring that existing users can get all information stored about them in an easy manner. Ensuring that existing users can revoke consent or exercise their right to be forgotten
- Any changes in consent need to be carefully audited
- Storing the current document version and allowing users to see to which version they consented
- If we update any of our existing terms and policies, we would like end users to re-consent to the new versions upon next login
- Retrieving only consented users for marketing and profiling tasks



Default Screen-Sets

- The suggested implementation is based on Customer Data Cloud's ***Update Profile Screen-Set***
- The screen-set includes several screens, including
  - ***Update Profile***
  - ***Privacy***: for reviewing the policies to which the user agreed
  - ***Communication***: for managing communication preferences

My Preference Center

WELCOME TO YOUR PREFERENCE CENTER

Privacy | Communication

First name: Joe

Last name: Bloggs

Email: Joe.Bloggs@gigya.com

Year of birth: 1981

Zip code: LDN18HX

Country: United Kingdom

Save

Change password

Privacy

MY LEGAL CONSENT

Back to My Preferences

Terms of Service: Active
Version of Consent: 3
Approved on 21/12/2017

Privacy: Active
Version of Consent: 1
Approved on 21/12/2017

☐ I agree to to website personalization

☑ I agree to have my email shared with 3rd parties

Save

Communication

MY COMMUNICATION PREFERENCES

Back to My Preferences

☐ Subscribed to the weekly email newsletter

☐ I want daily offers sent to my email

Save

Figure 208: Preference Center Implementation

Exercise 7

Figure 209: Exercise 7

LESSON SUMMARY

You should now be able to:

- Manage customer consent

UNIT 8 Extensibility

UNIT OBJECTIVES

- Use Webhooks, Extensions and JavaScript Parameters

Unit 8
Lesson 1

Understanding Extensibility

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use Webhooks, Extensions and JavaScript Parameters

Webhooks

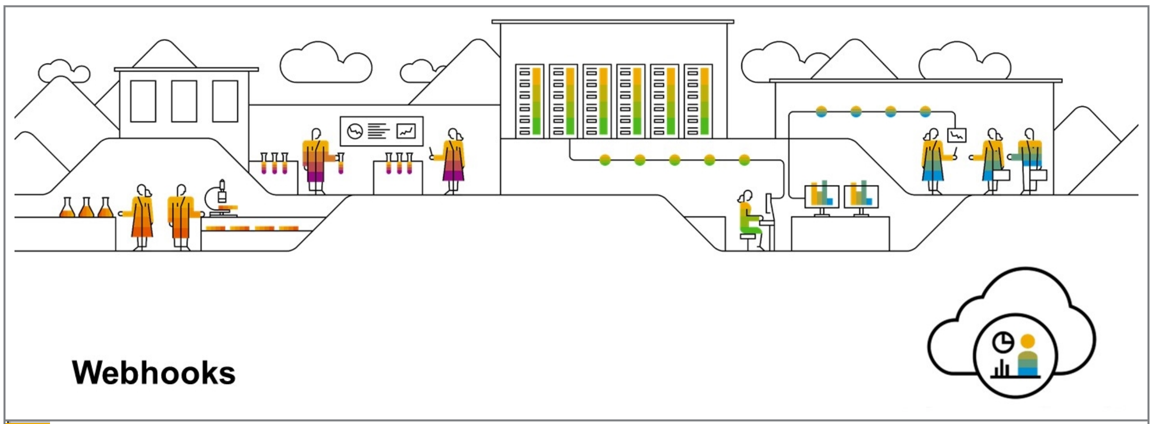
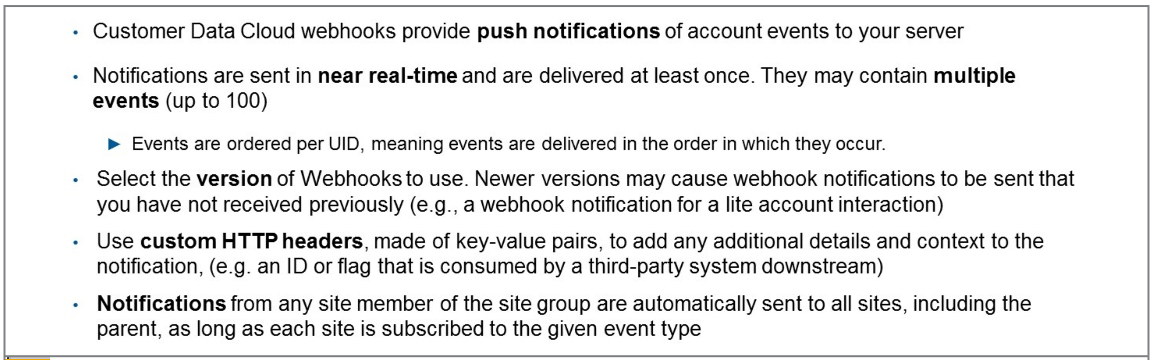


Figure 210: Webhooks



- Customer Data Cloud webhooks provide **push notifications** of account events to your server
- Notifications are sent in **near real-time** and are delivered at least once. They may contain **multiple events** (up to 100)
  - Events are ordered per UID, meaning events are delivered in the order in which they occur.
- Select the **version** of Webhooks to use. Newer versions may cause webhook notifications to be sent that you have not received previously (e.g., a webhook notification for a lite account interaction)
- Use **custom HTTP headers**, made of key-value pairs, to add any additional details and context to the notification, (e.g. an ID or flag that is consumed by a third-party system downstream)
- **Notifications** from any site member of the site group are automatically sent to all sites, including the parent, as long as each site is subscribed to the given event type

Figure 211: Webhooks Overview

1. Open http://console.gigya.com and go to ***Site Settings > Registration as a Service > Webhooks***
2. Enter a unique ***Name*** for the Webhook
3. Enter the ***Notification URL*** on your server to which webhook notification objects will be sent when the subscribed event is fired
4. Choose whether to ***Sign notifications using*** your *partner secret key* or using a *user/app key* pair
5. Select the ***Events*** to which this webhook is subscribed
6. Add any number of ***Custom Headers*** you want to post with your request
7. Select the ***Version*** of Webhook to use
8. Click on *Create* button

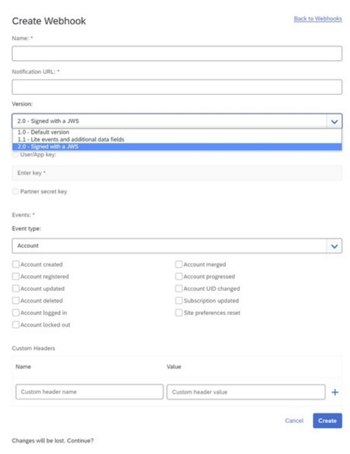


Figure 212: How to configure a Webhook

Custom Headers

You can add custom HTTP headers, made of key-value pairs, to be sent together with the webhook notification. You can use these headers to add any additional details and context to the notification, e.g. an ID or flag that is consumed by a third-party system downstream.

Or, just like everything else, you can use the REST API:

Accounts.webhooks.set

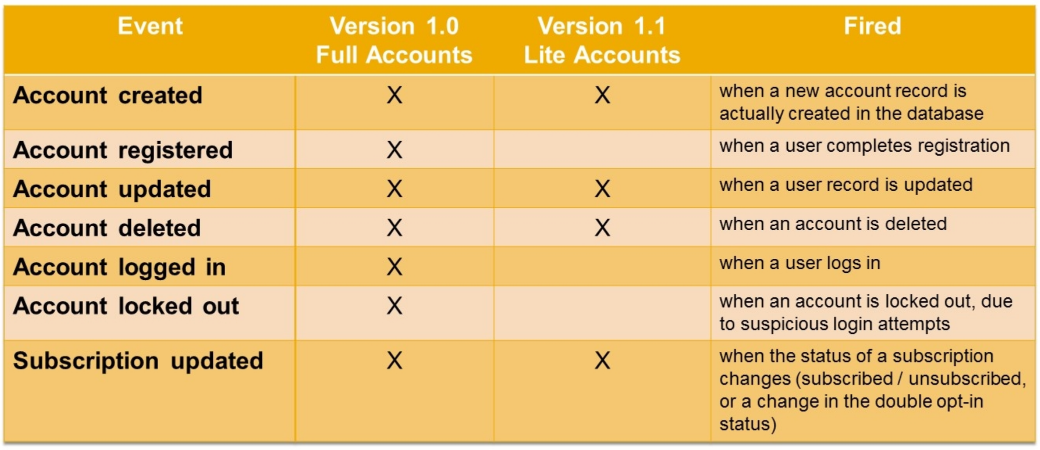
- Used to create a new webhook or update an existing one
- To update an existing webhook, pass the name of the webhook to update along with any relevant parameters. (URL, Events, Name, and Active are required)

Accounts.webhooks.getStatus

- - Used to check the status of a defined webhook, the time the last bulk was sent from servers and the status received for them from the downstream system.

Common Pitfalls:

- Passing a new name with an existing URL will return an error. To rename a webhook, first delete it, then create a new one
- Make sure you're using the proper Customer Data Cloud data center associated with the API key! (us1, au1, eu1, ru1, etc.)
- Must be sent over HTTPS



Event	Version 1.0 Full Accounts	Version 1.1 Lite Accounts	Fired
Account created	X	X	when a new account record is actually created in the database
Account registered	X		when a user completes registration
Account updated	X	X	when a user record is updated
Account deleted	X	X	when an account is deleted
Account logged in	X		when a user logs in
Account locked out	X		when an account is locked out, due to suspicious login attempts
Subscription updated	X	X	when the status of a subscription changes (subscribed / unsubscribed, or a change in the double opt-in status)

Figure 213: Supported events depending on the version and type of account

Current supported versions:

1.0 - Webhooks will be sent out only for interactions performed by fully registered accounts.

1.1 - Webhooks will be sent out also for lite account interactions, and will include a callId and version.

Account created: Note that when using version 1.1, if the account has progressed from lite to full, the account created event will not fire for a full registration, since the account has already been created

2.0 - Webhooks will be signed with a JWT and support Global Access. This version can be verified with the accounts.getJWTPublicKey endpoint without having to exchange secrets with the webhook recipient. It adds the API key of the site that initiated the notification to the webhook payload, supporting site group scenarios where an event should be triggered only when a webhook was initiated from a specific site

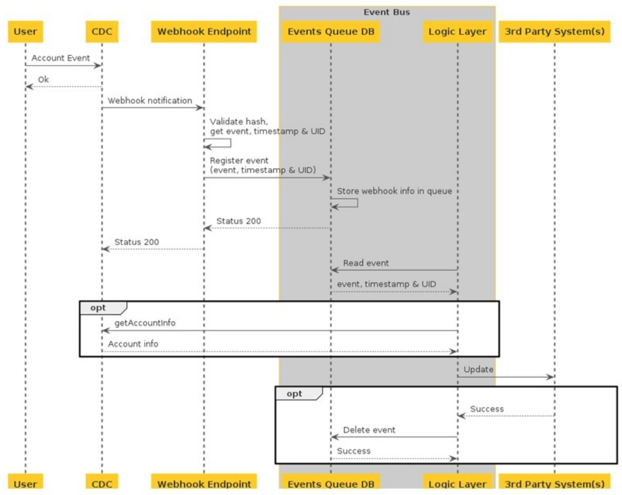


Figure 214: End-to-End WebHook: Sample Architecture

End-to-End WebHook: EndPoint Logic

Once a WebHook notifies the notification URL of an event, this endpoint needs to perform a few steps:

1. Verify the X-Gigya-HMAC-SHA1 header to ensure the authenticity of the notification

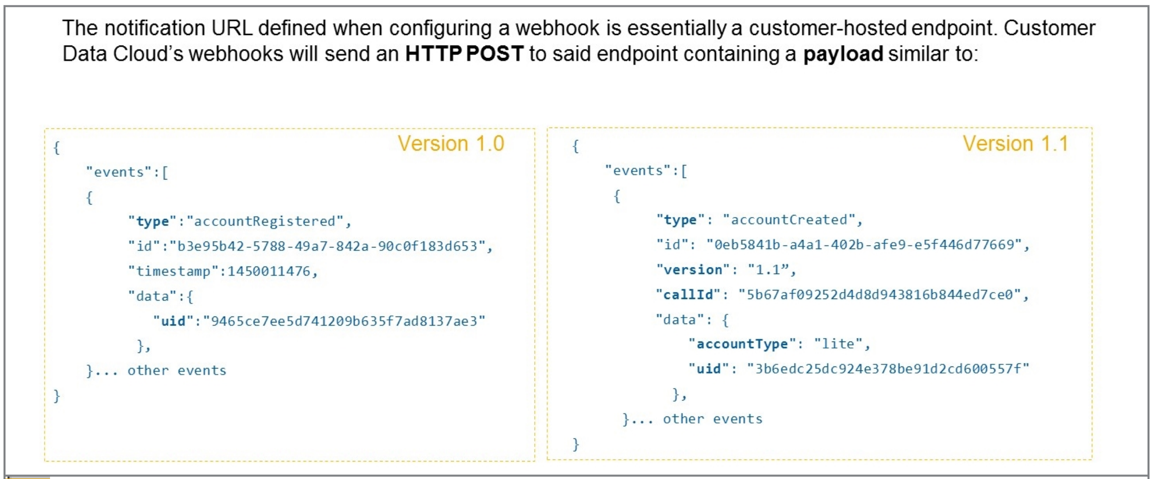
To verify the hash:

1. Create an HMAC-SHA1 hash of the JSON payload using a user secret or partner secret as the key. For more information on signing notifications using a user secret. See https://developers.gigya.com/display/GD/accounts.webhooks.set+REST
2. Compare your hash to the one received in the X-Gigya-HMAC-SHA1 header
3. Send a success response back to Customer Data Cloud

Customer Data Cloud considers notifications delivered if a 2xx Success HTTP status code is received (e.g., 200 OK, 201 Created, 202 Accepted). If we receive anything other than a Success response from your server, we will resend the

notification at increasing intervals up to one hour. If we receive no response at all, we assume that your callback endpoint is offline

1. Read the payload to retrieve the notification type and UID
2. Perform a REST call to accounts.getAccountInfo with the UID provided
3. Implement business logic



The notification URL defined when configuring a webhook is essentially a customer-hosted endpoint. Customer Data Cloud's webhooks will send an **HTTP POST** to said endpoint containing a **payload** similar to:

Version 1.0

```
{
    "events":[
    {
         "type":"accountRegistered",
         "id":"b3e95b42-5788-49a7-842a-90c0f183d653",
         "timestamp":1450011476,
         "data":{
            "uid":"9465ce7ee5d741209b635f7ad8137ae3"
          },
    }... other events
}
```

Version 1.1

```
{
    "events":[
     {
         "type": "accountCreated",
         "id": "0eb5841b-a4a1-402b-afe9-e5f446d77669",
         "version": "1.1",
         "callId": "5b67af09252d4d8d943816b844ed7ce0",
         "data": {
            "accountType": "lite",
            "uid": "3b6edc25dc924e378be91d2cd600557f"
          },
      }... other events
}
```

Figure 215: End-to-End WebHook: Notification URL

Extensions

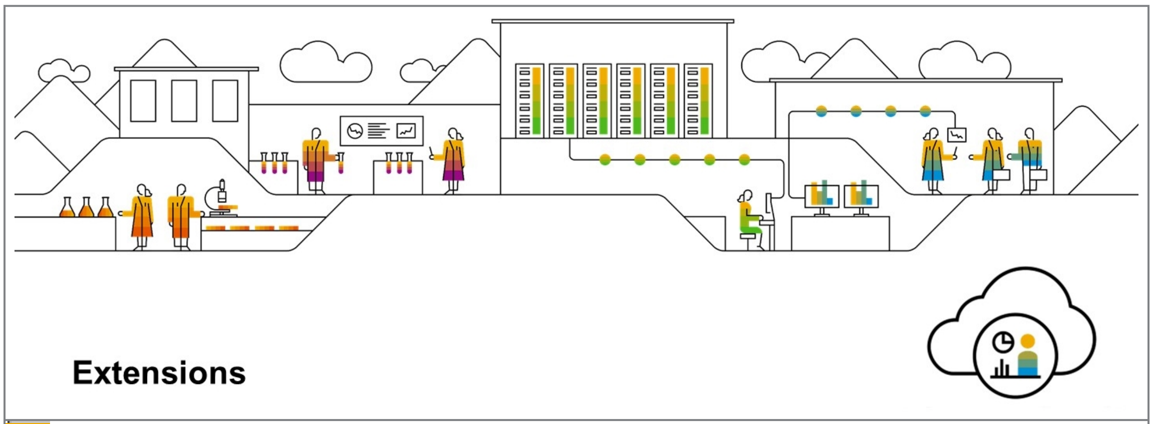


Figure 216: Extensions

- Extensions support secure, server-side, synchronous execution of your hosted custom code
- Extensions enable quickly implementing your custom data validations and restrictions
- Extension Service will connect CDC API endpoints with your code thru an HTTPS POST request
- What **API endpoints** are supported by Extensions?
  - **OnBeforeAccountsRegister** triggered within the accounts.register API, right after CDC runs all validation checks required for creating the user in the database and right before creating the user
  - **OnBeforeAccountsLogin** triggered during accounts.login, right after CDC runs the validation checks that are required for logging the user in
  - **OnBeforeSocialLogin** triggered during socialize.login, right after CDC runs the validation checks that are required for logging the user in
  - **OnBeforeSetAccountInfo** triggered during accounts.setAccountInfo, immediately before account data is written to the database

Figure 217: Extensions Overview

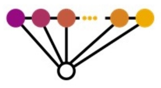
https://developers.gigya.com/display/GD/Extensions#Extensions-OnBeforeAccountsRegister

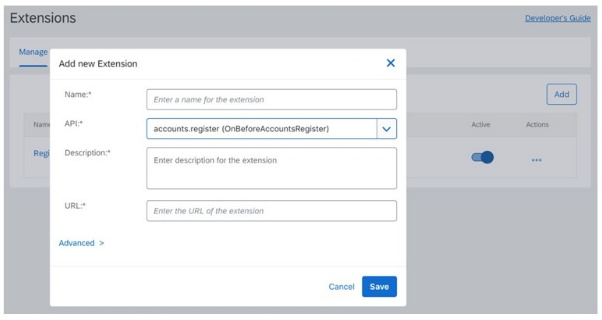
https://developers.gigya.com/display/GD/Extensions#Extensions-OnBeforeAccountsLogin

https://developers.gigya.com/display/GD/Extensions#Extensions-OnBeforeSocialLogin

https://developers.gigya.com/display/GD/Extensions#Extensions-OnBeforeSetAccountInfo

1. Open console.gigya.com
2. **Select a site,** go to ***Settings > Basic > Extensions***
3. Click on **Add New Extension**
4. Enter a **uniqueName**
5. Select the **API endpoint** that will trigger the Extension
6. Enter the **HTTPS POST URL** on your server to which JSON objects will be sent
7. add **Advanced parameters** like *Timeout, fallback policy,* or *Integration* (if needed)
8. Click on Save





Extensions

Add new Extension

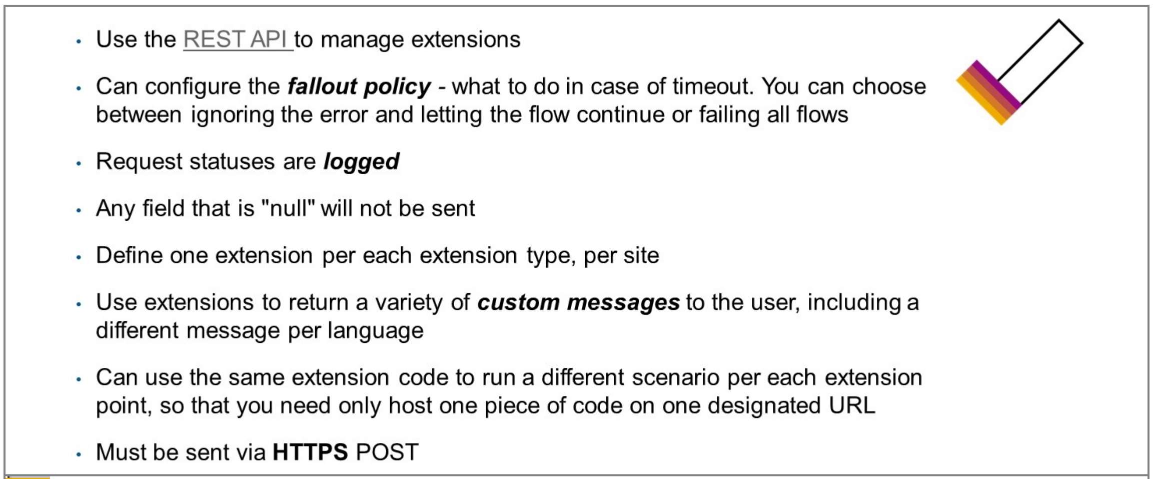
Name:* Enter a name for the extension

API:* accounts.register (OnBeforeAccountsRegister)

Description:* Enter description for the extension

URL:* Enter the URL of the extension

Figure 218: How to configure a new Extension



- Use the REST API to manage extensions
- Can configure the ***fallout policy*** - what to do in case of timeout. You can choose between ignoring the error and letting the flow continue or failing all flows
- Request statuses are ***logged***
- Any field that is "null" will not be sent
- Define one extension per each extension type, per site
- Use extensions to return a variety of ***custom messages*** to the user, including a different message per language
- Can use the same extension code to run a different scenario per each extension point, so that you need only host one piece of code on one designated URL
- Must be sent via **HTTPS** POST

Figure 219: Extensions Features

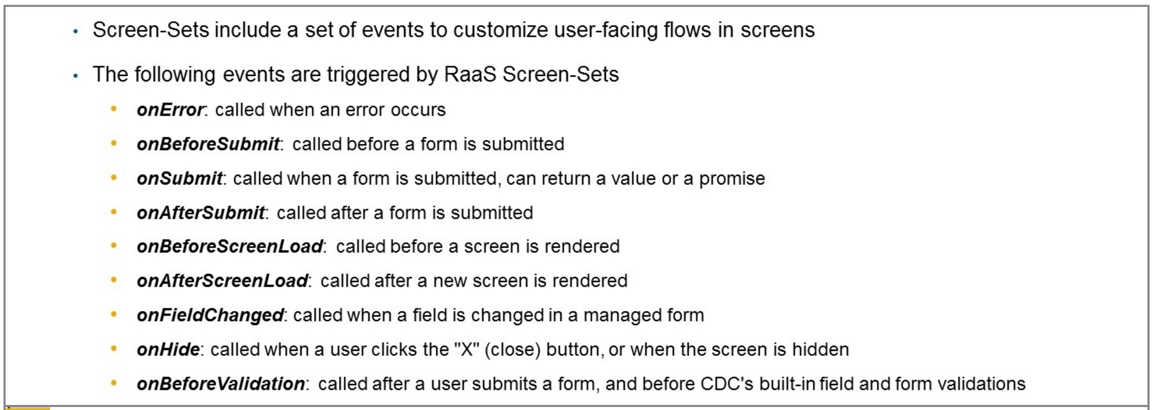
Posting Requests and Receiving the Response

https://developers.gigya.com/display/GD/Extensions#Extensions-ExtensionCode

For more details, watch the following video: https://enable.cx.sap.com/media/1_ucyxrfoj

JavaScript Parameters

Figure 220: JavaScript Parameters

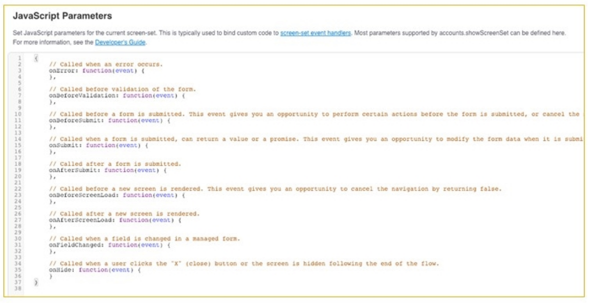


- Screen-Sets include a set of events to customize user-facing flows in screens
- The following events are triggered by RaaS Screen-Sets
  - ***onError***: called when an error occurs
  - ***onBeforeSubmit***: called before a form is submitted
  - ***onSubmit***: called when a form is submitted, can return a value or a promise
  - ***onAfterSubmit***: called after a form is submitted
  - ***onBeforeScreenLoad***: called before a screen is rendered
  - ***onAfterScreenLoad***: called after a new screen is rendered
  - ***onFieldChanged***: called when a field is changed in a managed form
  - ***onHide***: called when a user clicks the "X" (close) button, or when the screen is hidden
  - ***onBeforeValidation***: called after a user submits a form, and before CDC's built-in field and form validations

Figure 221: JavaScript Parameters Overview

https://developers.gigya.com/display/GD/JavaScript+Parameters

1. Open console.gigya.com
2. Go to *Settings* > *Registration-as-a-Service* > *Screen-Sets*
3. Open the related screen-set and screen
4. Select ***JavaScript Parameters***
5. Select the **event** that you want to customize on the client side
6. Add your custom JavaScript client code
7. Click Save.



JavaScript Parameters

Set JavaScript parameters for the current screen-set. This is typically used to bind custom code to screen-set event handlers. Most parameters supported by accounts.showScreenSet can be defined here.
For more information, see the Developer's Guide.

```
{
    // Called when an error occurs.
    onError: function(event) {
    },

    // Called before validation of the form.
    onBeforeValidation: function(event) {
    },

    // Called before a form is submitted. This event gives you an opportunity to perform certain actions before the form is submitted, or cancel the
    onBeforeSubmit: function(event) {
    },

    // Called when a form is submitted, can return a value or a promise. This event gives you an opportunity to modify the form data when it is submi
    onSubmit: function(event) {
    },

    // Called after a form is submitted.
    onAfterSubmit: function(event) {
    },

    // Called before a new screen is rendered. This event gives you an opportunity to cancel the navigation by returning false.
    onBeforeScreenLoad: function(event) {
    },

    // Called after a new screen is rendered.
    onAfterScreenLoad: function(event) {
    },

    // Called when a field is changed in a managed form.
    onFieldChanged: function(event) {
    },

    // Called when a user clicks the "X" (close) button or the screen is hidden following the end of the flow.
    onHide: function(event) {
    }
}
```

Figure 222: How to use JavaScript Parameters

```
{ // JavaScript Parameters JSON Obj
    // Called when a field is changed in a managed form.
    onFieldChanged: function(event) {
        window.eye = (typeof(window.eye) == 'undefined') ? 0 :  window.eye +1;
        console.log("field changed: " + window.eye);
    }
}
```

Figure 223: Example of coding JavaScript event handlers

	Webhooks	Extensions	JavaScript Parameters
Server / Client	Asynchronous, server-to-server	Synchronous, server-to-server	Client-side
What are the triggers?	After a specific event occurs	Within the CDC flow	Client-side screen-set triggers
When?	Post-event	Real-time	Real-time
Event type	CDC webhooks	Extension endpoints	Screen-set events
Useful for	Downstream flows and application	Real-time decision making	Real-time response to user's input

Figure 224: Extensibility Techniques Comparison

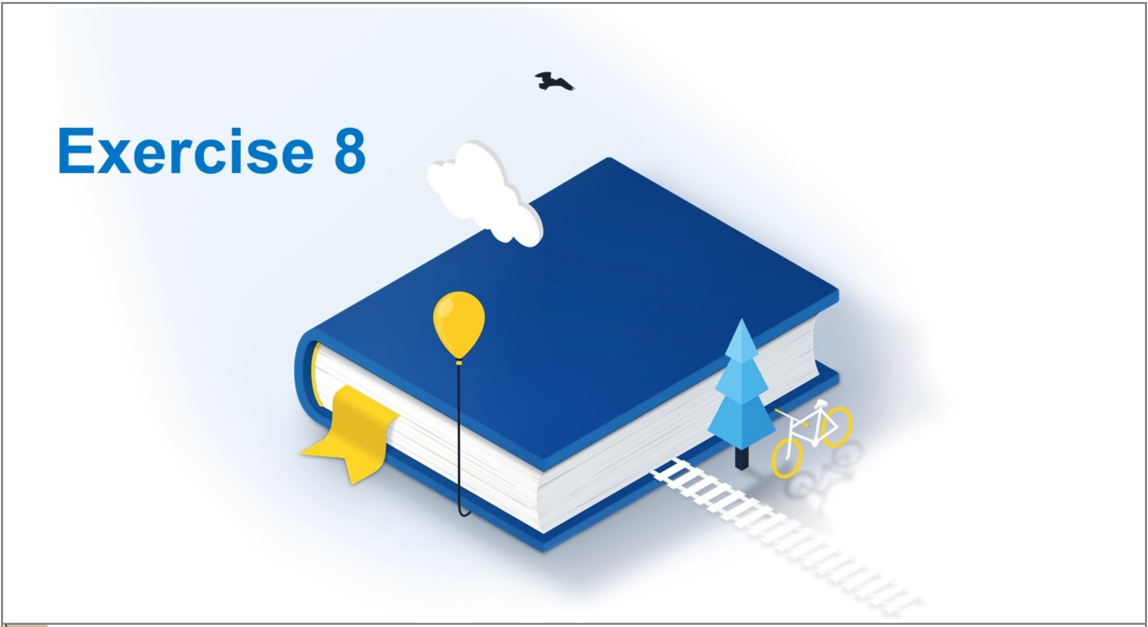


Figure 225: Exercise 8

LESSON SUMMARY

You should now be able to:

- Use Webhooks, Extensions and JavaScript Parameters

UNIT 9 Identity Exchange

UNIT OBJECTIVES

- Use SAP Customer Profile, GConnectors, and IdentitySync

Unit 9
Lesson 1

Understanding Identity Exchange

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use SAP Customer Profile, GConnectors, and IdentitySync

SAP Customer Profile

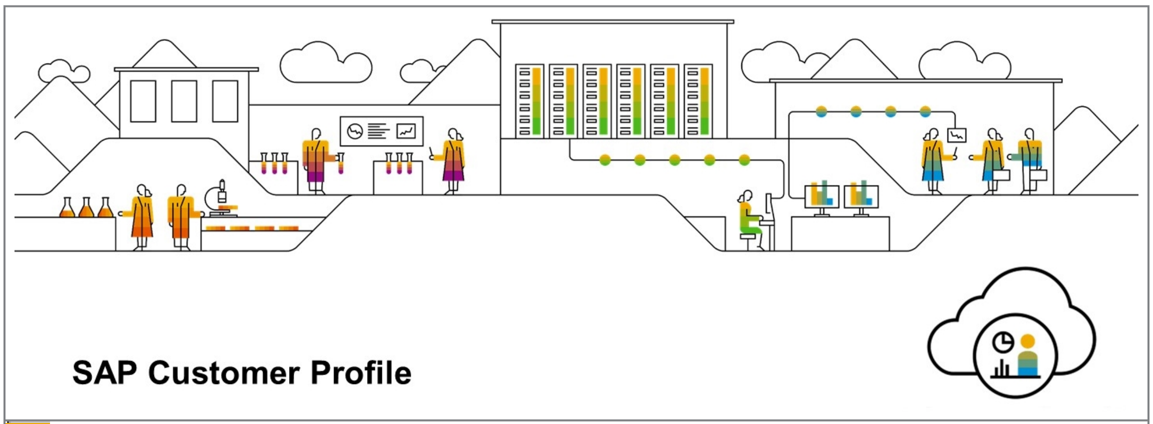
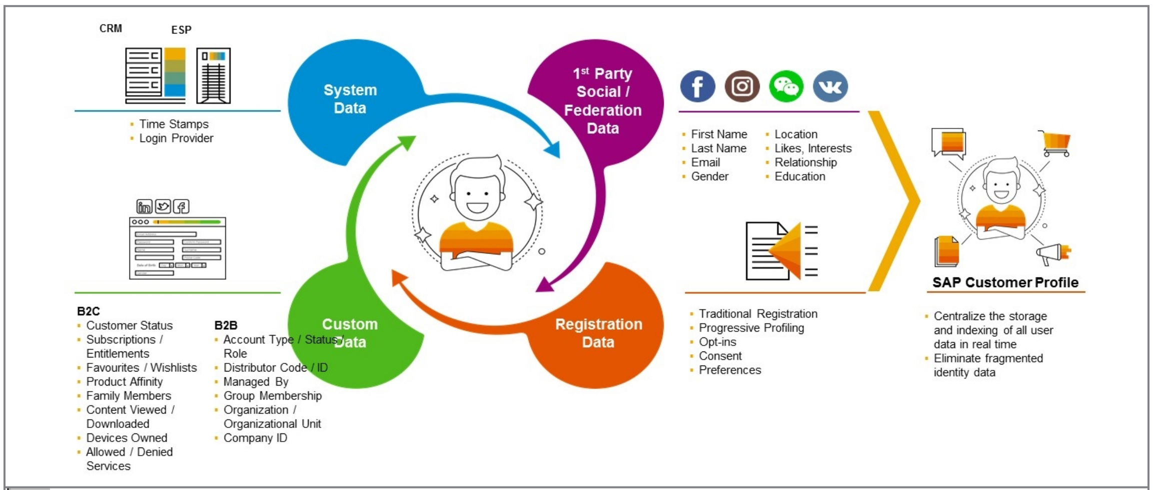


Figure 226: SAP Customer Profile



- Time Stamps
- Login Provider

B2C
- Customer Status
- Subscriptions / Entitlements
- Favourites / Wishlists
- Product Affinity
- Family Members
- Content Viewed / Downloaded
- Devices Owned
- Allowed / Denied Services

B2B
- Account Type / Status / Role
- Distributor Code / ID
- Managed By
- Group Membership
- Organization / Organizational Unit
- Company ID

- First Name
- Last Name
- Email
- Gender
- Location
- Likes, Interests
- Relationship
- Education

- Traditional Registration
- Progressive Profiling
- Opt-ins
- Consent
- Preferences

- Centralize the storage and indexing of all user data in real time
- Eliminate fragmented identity data

Figure 227: SAP Customer Profile - Flexible Account Structure

The Customer Data Cloud database leverages Big Data technologies to store user data in a very flexible and efficient way

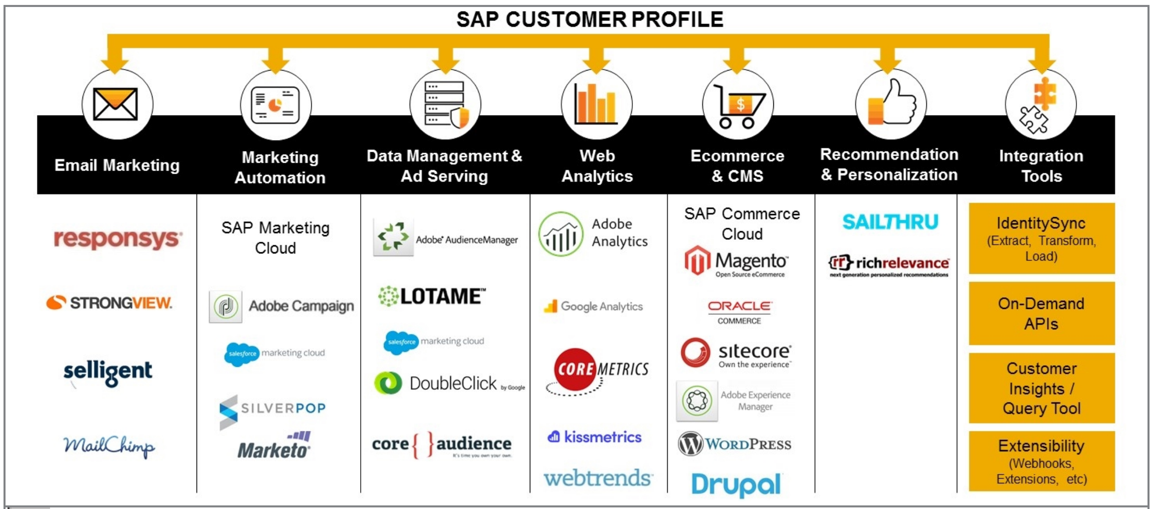


Figure 228: Profile - Transform, Orchestrate, and Govern Identity & Profiles

https://developers.gigya.com/display/GD/Identity+Exchange+-+Partner+Integrations

Customer Data Cloud's technology is agile and flexible. Our products can be integrated with Marketing Cloud, analytics, CRM, ESP, CMS, and open platforms. We design products and APIs to leverage and enhance our customers' existing services.

GConnectors

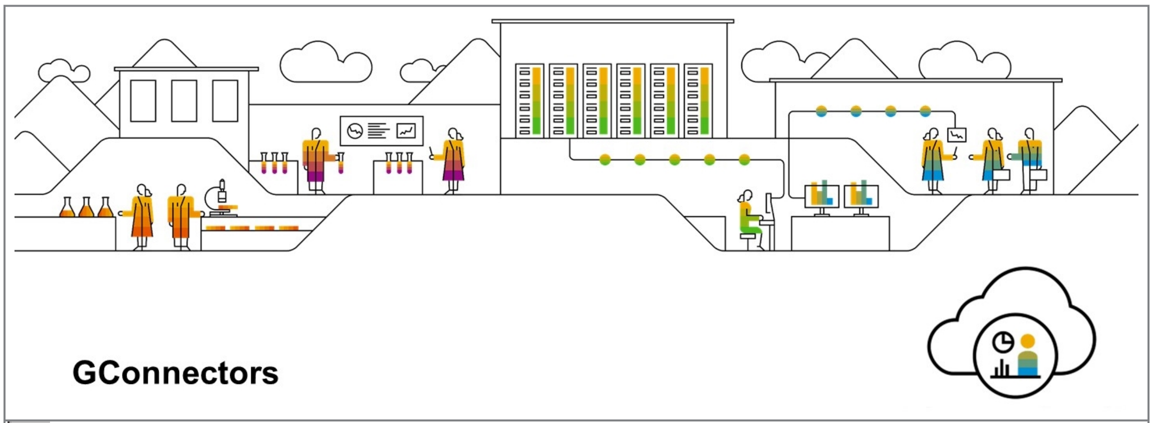
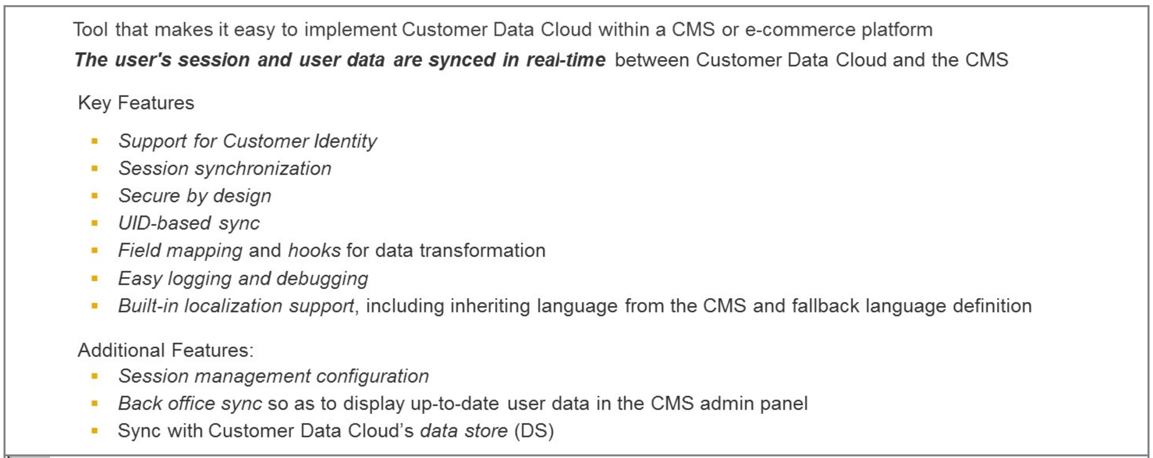


Figure 229: GConnectors



Tool that makes it easy to implement Customer Data Cloud within a CMS or e-commerce platform

The user's session and user data are synced in real-time between Customer Data Cloud and the CMS

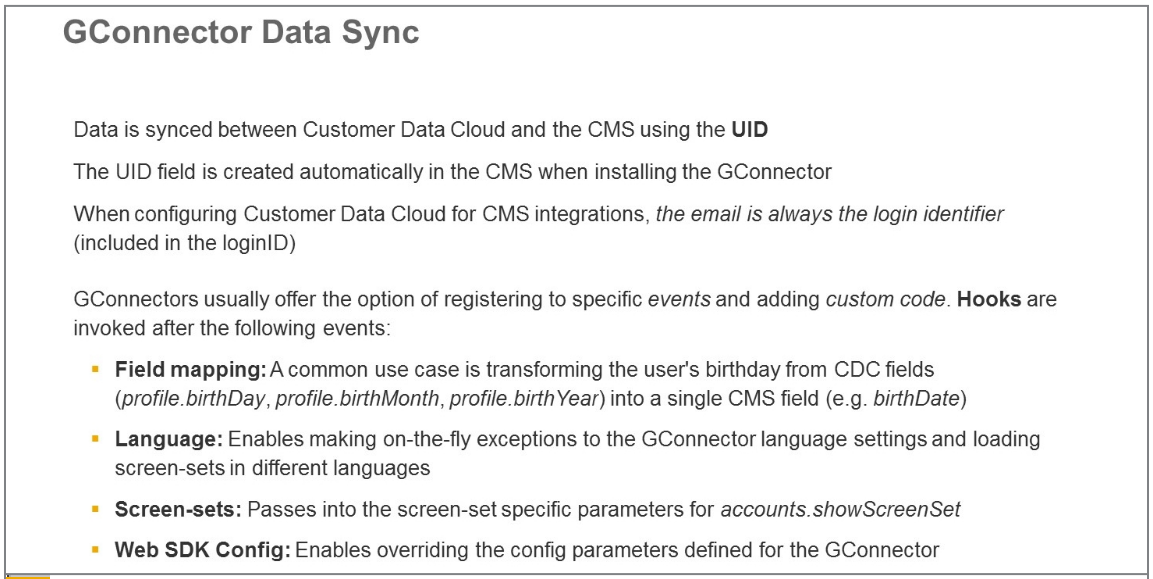
Key Features

- *Support for Customer Identity*
- *Session synchronization*
- *Secure by design*
- *UID-based sync*
- *Field mapping* and *hooks* for data transformation
- *Easy logging and debugging*
- *Built-in localization support*, including inheriting language from the CMS and fallback language definition

Additional Features:

- *Session management configuration*
- *Back office sync* so as to display up-to-date user data in the CMS admin panel
- Sync with Customer Data Cloud's *data store* (DS)

Figure 230: GConnector for CMS / eCommerce



GConnector Data Sync

Data is synced between Customer Data Cloud and the CMS using the **UID**

The UID field is created automatically in the CMS when installing the GConnector

When configuring Customer Data Cloud for CMS integrations, *the email is always the login identifier* (included in the loginID)

GConnectors usually offer the option of registering to specific *events* and adding *custom code*. **Hooks** are invoked after the following events:

- **Field mapping:** A common use case is transforming the user's birthday from CDC fields (*profile.birthDay*, *profile.birthMonth*, *profile.birthYear*) into a single CMS field (e.g. *birthDate*)
- **Language:** Enables making on-the-fly exceptions to the GConnector language settings and loading screen-sets in different languages
- **Screen-sets:** Passes into the screen-set specific parameters for *accounts.showScreenSet*
- **Web SDK Config:** Enables overriding the config parameters defined for the GConnector

Figure 231: GConnectors

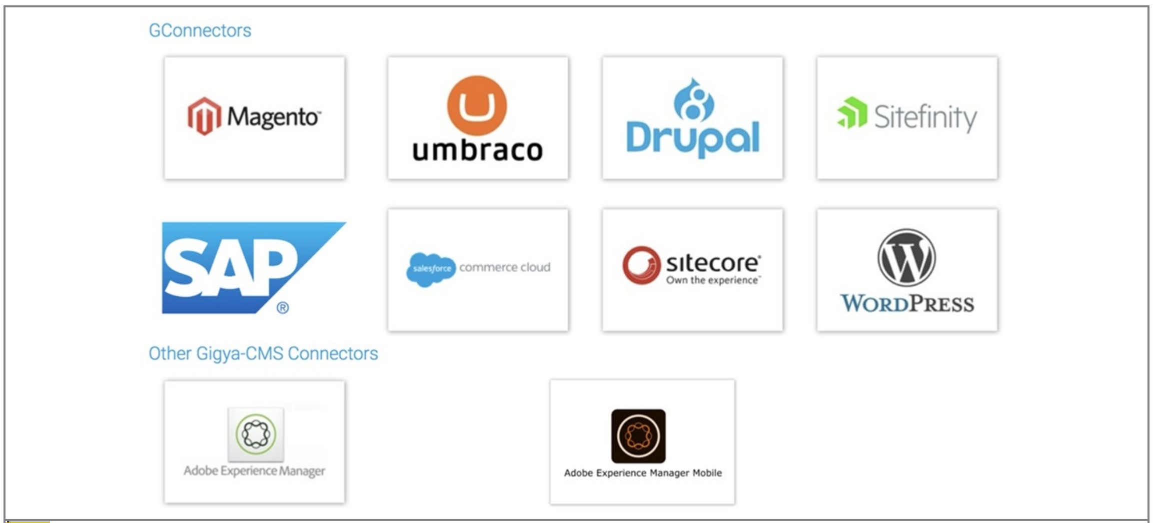

GConnectors
Magento
umbraco
Drupal
Sitefinity
SAP
salesforce commerce cloud
sitecore Own the experience
WordPress
Other Gigya-CMS Connectors
Adobe Experience Manager
Adobe Experience Manager Mobile

Figure 232: Supported GConnectors

https://developers.gigya.com/display/GD/GConnector+-+CMS+and+E-Commerce+Integrations#GConnector-CMSandE-CommerceIntegrations-GConnectors

Offline synchronization was added to Magento 2, Drupal 8, and WordPress, allowing better privacy compliance due to periodic user consent data replication between CDC and these platforms

Support for AEM Touch UI

New UI for WordPress GConnector

All Sitecore GConnectors prior to 9.0 have been deprecated and are no longer available

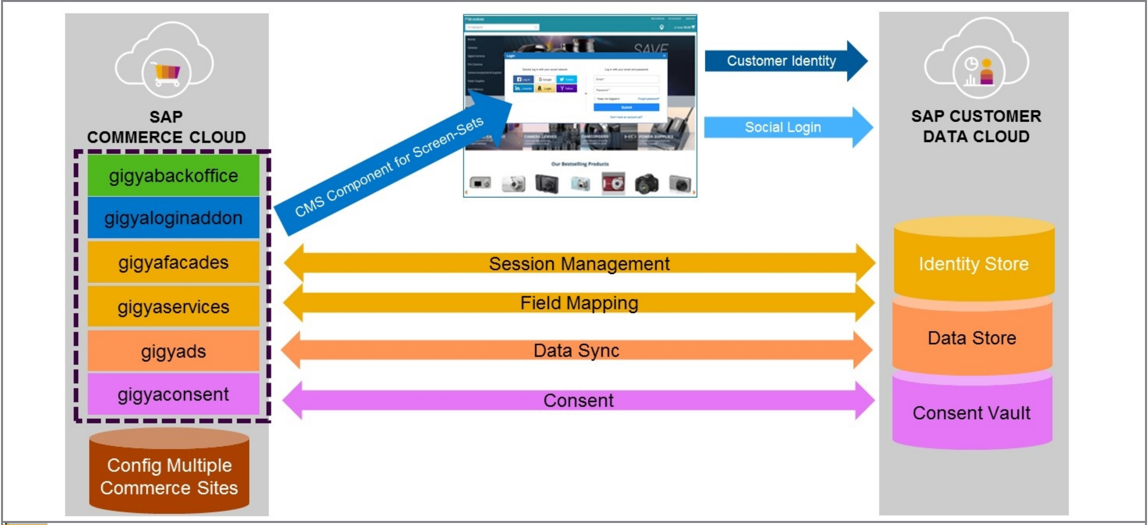


Figure 233: GConnector for SAP Commerce Cloud

SAP Customer Data Cloud integration provides a set of extensions to enable registration-as-a-service and to manage profile updates and user consent from SAP Commerce.

IdentitySync

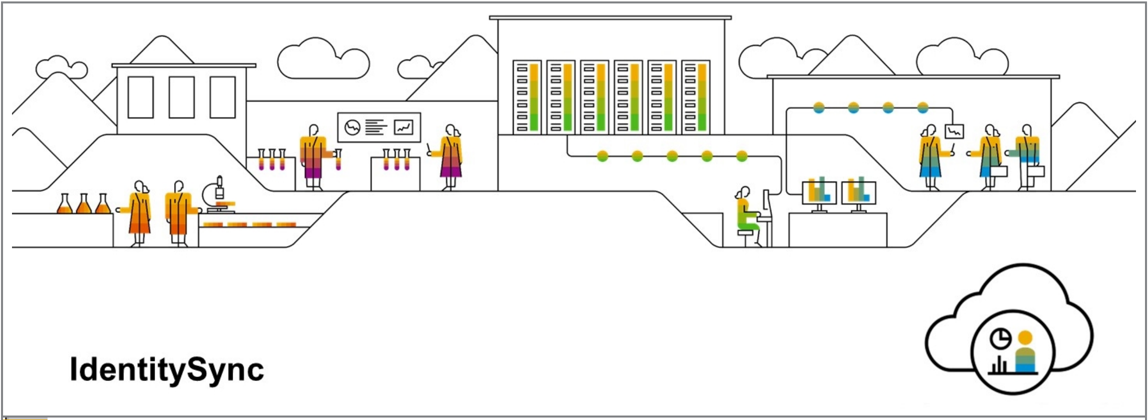
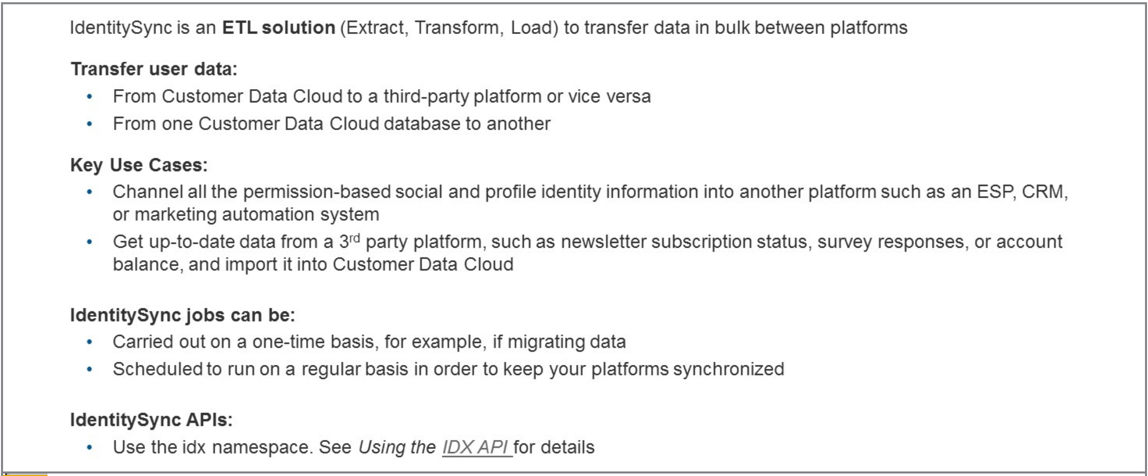


Figure 234: IdentitySync



IdentitySync is an **ETL solution** (Extract, Transform, Load) to transfer data in bulk between platforms

Transfer user data:

- From Customer Data Cloud to a third-party platform or vice versa
- From one Customer Data Cloud database to another

Key Use Cases:

- Channel all the permission-based social and profile identity information into another platform such as an ESP, CRM, or marketing automation system
- Get up-to-date data from a 3rd party platform, such as newsletter subscription status, survey responses, or account balance, and import it into Customer Data Cloud

IdentitySync jobs can be:

- Carried out on a one-time basis, for example, if migrating data
- Scheduled to run on a regular basis in order to keep your platforms synchronized

IdentitySync APIs:

- Use the idx namespace. See *Using the IDX API* for details

Figure 235: IdentitySync

https://developers.gigya.com/display/GD/IdentitySync#

New parameters in datasource.read.gigya.audit allow using IdentitySync to export audit log items from a different SAP Customer Data Cloud site.

A new notifyLastRecord parameter was added to the record.evaluate script used for creating custom scripts. This is used to indicate that the last record in the batch has been handled.

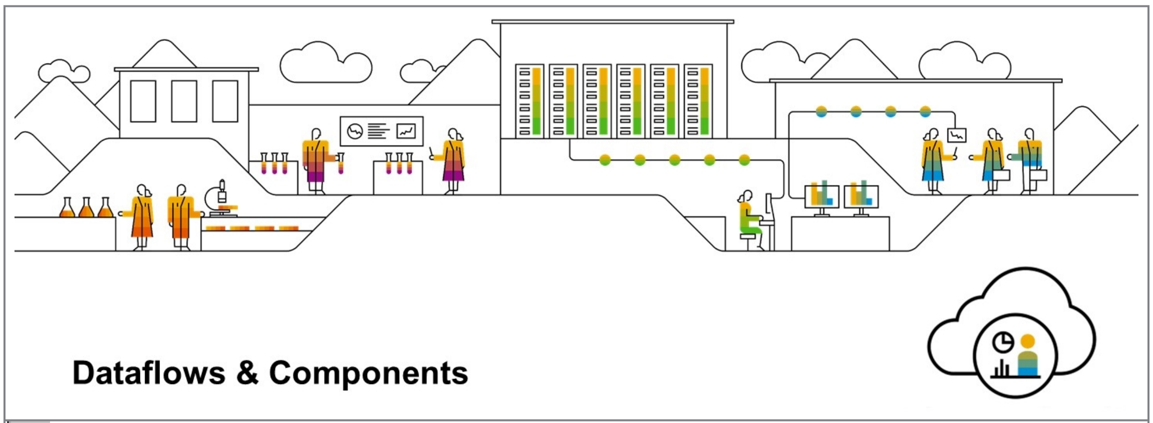
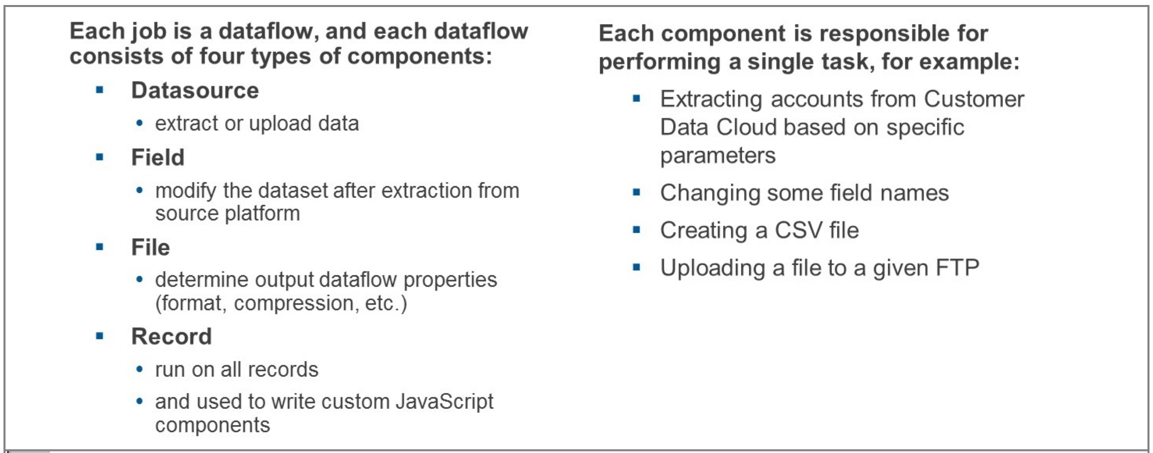


Figure 236: IdentitySync



Each job is a dataflow, and each dataflow consists of four types of components:

- **Datasource**
  - extract or upload data
- **Field**
  - modify the dataset after extraction from source platform
- **File**
  - determine output dataflow properties (format, compression, etc.)
- **Record**
  - run on all records
  - and used to write custom JavaScript components

Each component is responsible for performing a single task, for example:

- Extracting accounts from Customer Data Cloud based on specific parameters
- Changing some field names
- Creating a CSV file
- Uploading a file to a given FTP

Figure 237: IdentitySync Dataflows

For detailed information, visit the Component Repository (https://developers.gigya.com/display/GD/Component+Repository) and see sample dataflow templates (https://developers.gigya.com/display/GD/Dataflow+Templates).

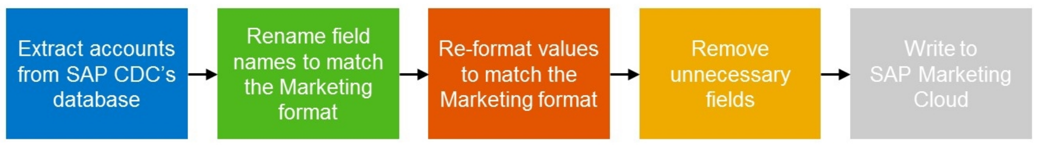

Extract accounts from SAP CDC's database
Rename field names to match the Marketing format
Re-format values to match the Marketing format
Remove unnecessary fields
Write to SAP Marketing Cloud

Figure 238: SAP Marketing Cloud Dataflow Example

Supported Use-Cases

- Campaign management
- Sync lite and full accounts (see Lite Registration section)
- Sync communication preferences data

- Sync consent status
- Segment users in SAP Marketing Cloud based on 1st-party data from CDC
  - Based on segmentation targeted ads
  - Social network campaigns
  - Personalization
  - Content recommendation
  - Look-alike audience
  - DMP integration

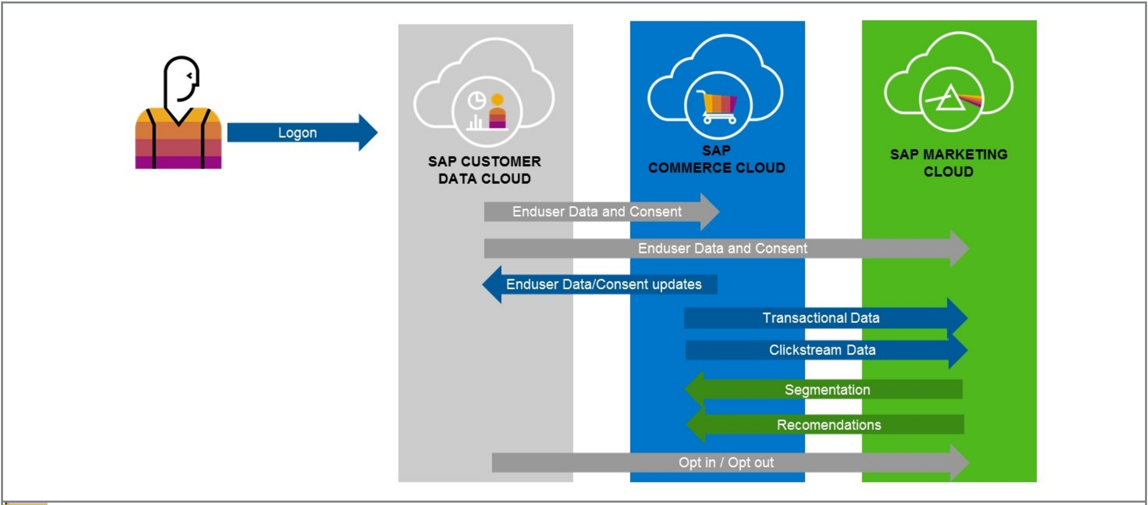


Figure 239: Integrating three caces using a common UID for the end-user

In websites that use both Commerce and Marketing, information can be channeled between the platforms through CDC, based on the common CDC UID of the end-user.

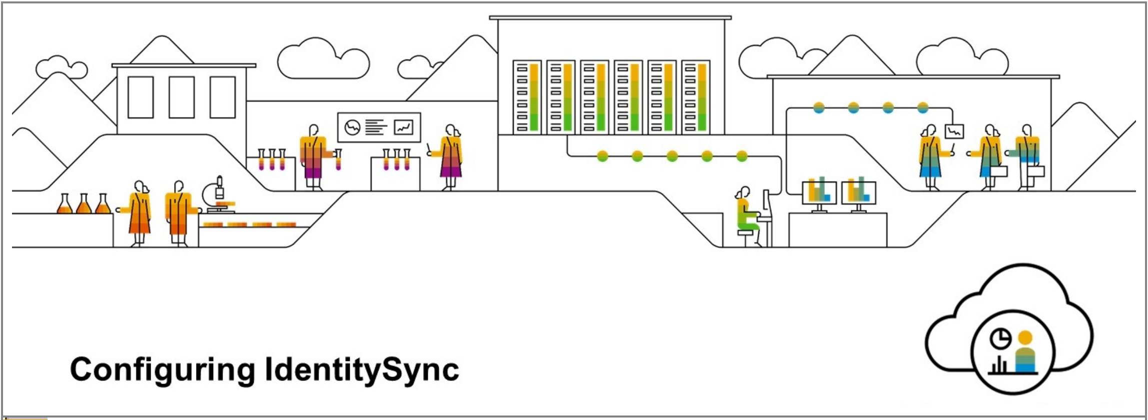
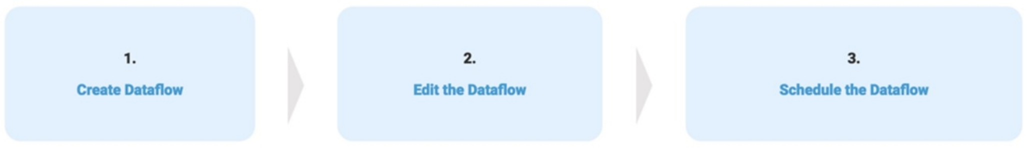


Figure 240: IdentitySync



- Dataflow templates

- Credentials to access each system
- Identity Studio
- Add/Edit components
- Handle failed records

- All jobs are scheduled in UTC timezone

Figure 241: Steps to Configure IdentitySync

For more details: https://developers.gigya.com/display/GD/IdentitySync

There are four different **types** of components:

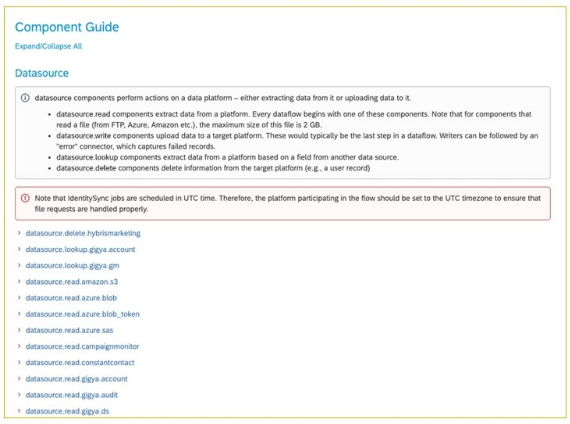
1. **datasource** components perform actions on a data platform – either extracting data from it or uploading data to it
   - **datasource.read** components extract data from a platform
   - **datasource.write** components upload data to a target platform
   - **datasource.lookup** components extract data from a platform based on a field from another data source
   - **datasource.delete** components delete information from the target platform
2. **field** components modify the dataset in various ways after it is extracted from the source platform
3. **file** components determine the format, file compression, and other properties of the output of the dataflow before it is uploaded to the target platform
4. **record** components run on all records. These are used when creating custom components that use JavaScript

Figure 242: IdentitySync Components

- A link to the IdentitySync Component Guide can be found in the Customer Data Cloud Developer's Guide:

  https://developers.gigya.com/display/GD/Component+Repository

- The documentation is detailed and describes the components available for use in an IdentitySync job



Component Guide

Expand/Collapse All

Datasource

ⓘ datasource components perform actions on a data platform – either extracting data from it or uploading data to it.

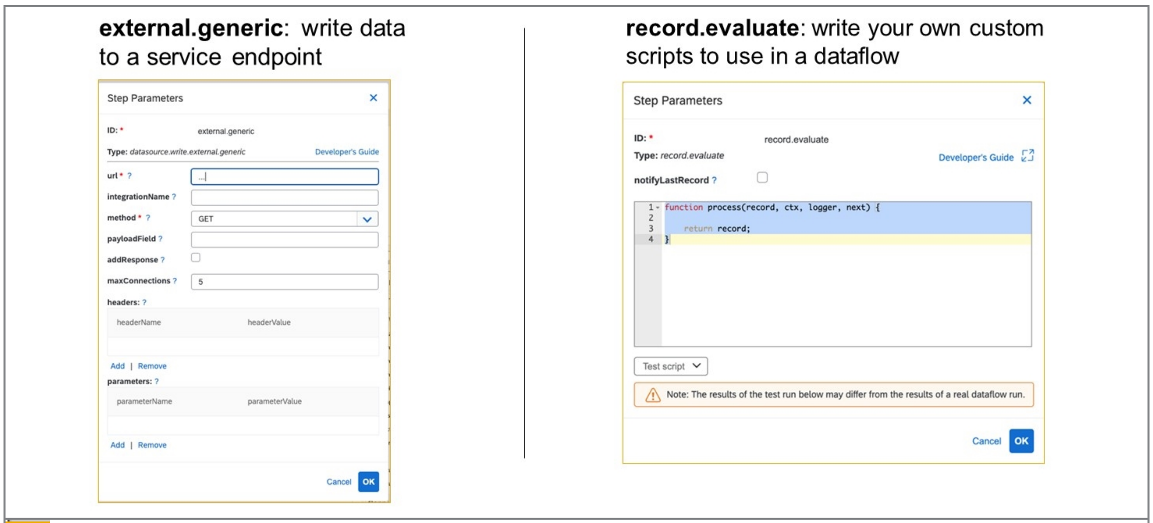
- datasource.read components extract data from a platform. Every dataflow begins with one of these components. Note that for components that read a file (from FTP, Azure, Amazon etc.), the maximum size of this file is 2 GB.
- datasource.write components upload data to a target platform. These would typically be the last step in a dataflow. Writers can be followed by an "error" connector, which captures failed records.
- datasource.lookup components extract data from a platform based on a field from another data source.
- datasource.delete components delete information from the target platform (e.g., a user record)

ⓘ Note that IdentitySync jobs are scheduled in UTC time. Therefore, the platform participating in the flow should be set to the UTC timezone to ensure that file requests are handled properly.

› datasource.delete.hybrismarketing
› datasource.lookup.gigya.account
› datasource.lookup.gigya.gm
› datasource.read.amazon.s3
› datasource.read.azure.blob
› datasource.read.azure.blob_token
› datasource.read.azure.sas
› datasource.read.campaignmonitor
› datasource.read.constantcontact
› datasource.read.gigya.account
› datasource.read.gigya.audit
› datasource.read.gigya.ds

Figure 243: Component Repository

https://developers.gigya.com/display/GD/Component+Repository



external.generic: write data to a service endpoint

Step Parameters

ID: * external.generic

Type: *datasource.write.external.generic* Developer's Guide

url * ?	...
integrationName ?	
method * ?	GET
payloadField ?	
addResponse ?	☐
maxConnections ?	5

headers: ?

headerName	headerValue

Add | Remove

parameters: ?

parameterName	parameterValue

Add | Remove

Cancel OK

record.evaluate: write your own custom scripts to use in a dataflow

Step Parameters

ID: * record.evaluate

Type: *record.evaluate* Developer's Guide

notifyLastRecord ? ☐

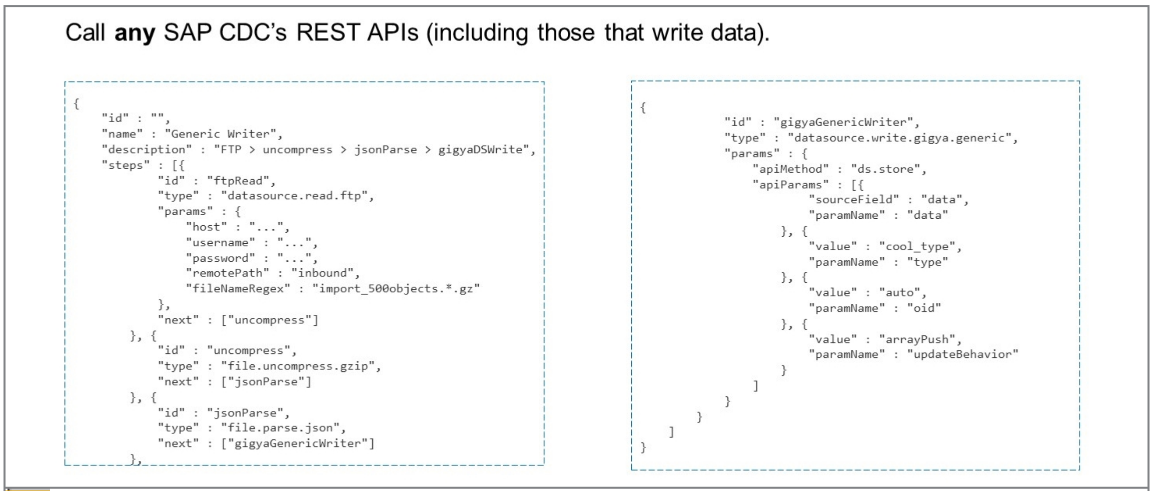
```
function process(record, ctx, logger, next) {

    return record;
}
```

Test script

Note: The results of the test run below may differ from the results of a real dataflow run.

Cancel OK

Figure 244: Relevant components

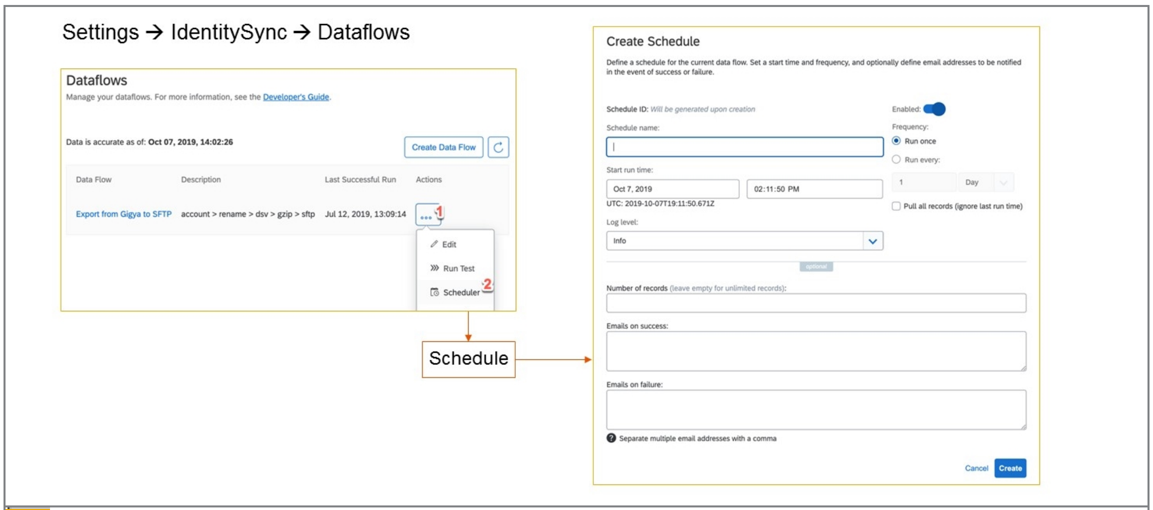


Call **any** SAP CDC’s REST APIs (including those that write data).

```
{
    "id" : "",
    "name" : "Generic Writer",
    "description" : "FTP > uncompress > jsonParse > gigyaDSWrite",
    "steps" : [{
            "id" : "ftpRead",
            "type" : "datasource.read.ftp",
            "params" : {
                "host" : "...",
                "username" : "...",
                "password" : "...",
                "remotePath" : "inbound",
                "fileNameRegex" : "import_500objects.*.gz"
            },
            "next" : ["uncompress"]
        }, {
            "id" : "uncompress",
            "type" : "file.uncompress.gzip",
            "next" : ["jsonParse"]
        }, {
            "id" : "jsonParse",
            "type" : "file.parse.json",
            "next" : ["gigyaGenericWriter"]
        },
```

```
{
            "id" : "gigyaGenericWriter",
            "type" : "datasource.write.gigya.generic",
            "params" : {
                "apiMethod" : "ds.store",
                "apiParams" : [{
                        "sourceField" : "data",
                        "paramName" : "data"
                    }, {
                        "value" : "cool_type",
                        "paramName" : "type"
                    }, {
                        "value" : "auto",
                        "paramName" : "oid"
                    }, {
                        "value" : "arrayPush",
                        "paramName" : "updateBehavior"
                    }
                ]
            }
        }
    ]
}
```

Figure 245: Relevant components: gigya.generic



Settings → IdentitySync → Dataflows

Dataflows

Manage your dataflows. For more information, see the Developer's Guide.

Data is accurate as of: **Oct 07, 2019, 14:02:26**

Create Data Flow

Data Flow	Description	Last Successful Run	Actions
Export from Gigya to SFTP	account > rename > dsv > gzip > sftp	Jul 12, 2019, 13:09:14	...

Create Schedule

Define a schedule for the current data flow. Set a start time and frequency, and optionally define email addresses to be notified in the event of success or failure.

Schedule ID: *Will be generated upon creation*

Enabled: ●

Schedule name:

Frequency: [✓ Run once] Run every: 1 Day

Start run time: Oct 7, 2019 02:11:50 PM

UTC: 2019-10-07T19:11:50.671Z

☐ Pull all records (ignore last run time)

Log level: Info

optional

Number of records (leave empty for unlimited records):

Emails on success:

Emails on failure:

Separate multiple email addresses with a comma

Cancel Create

Figure 246: Creating a Schedule

1. Under Actions, click the three dots, then click Scheduler.

2. Click Create Schedule.

3. Configure the schedule:
4. Enter a schedule name
5. Change the start time as needed
6. Choose whether to run once or at scheduled intervals
7. "Pull all records" should usually be selected only in the first run, when migrating records from one database to the other, and in any case should be used with caution. If the checkbox is not selected, and this is the first time this dataflow is run, records will be pulled according to the following logic:
   a. If the dataflow is set to run once, all records from the last 24 hours will be pulled.
   b. If the dataflow is recurring, records will be pulled according to the defined frequency. For example, if the dataflow is set to run once a week, the first time it is run, it will pull all records from the previous week.
8. (Optional) Enter the email adress(es) for success and failure of the dataflow run.
9. (Optional) Limit the run to a specific number of records. This is usually used for test runs. When running a test from the dashboard, a one-time schedule is created which runs immediately for 10 records.
10. Click Create and once you are back in the Schedule dashboard, click the Refresh button.
11. The status of the scheduling is indicated in the Status column.
12. You can stop a job mid-run by clicking the Stop icon under Actions.

Creating a Schedule through API

Scheduling Object

In the IDX API used by IdentitySync, a scheduling object defines the date, time,and frequency at which to run a given dataflow.

The idx.createScheduling REST method schedules a dataflow for execution.

Request URL

http[s]://idx.<Data_Center_ID>.gigya.com/idx.createScheduling.

Also, check the request payload here: https://developers.gigya.com/display/GD/Scheduling +Object

IdentitySync includes a built-in capability for separating failed records and writing them to a file, so that they may be reviewed and handled, and fed back into the flow.

How to add error handling to the workflow?

After a **writer** step, add an additional step that writes the records that did not complete the flow successfully into a separate file for review.

Under **Connector Type**, select the "Error path" connector.

Draw a connection from the original writer, to which successful records will be written, to the next step that handles failed records

```
{
            "id": "remove",
            "type": "field.remove",
            "params": {
                        "fields": ["profile","data"]
            },
            "next": [ "mailchimp"],
            "error": ["sftp"]
}
```

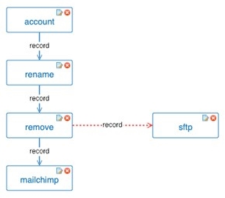


Figure 247: Handling Errors

Links for more details:

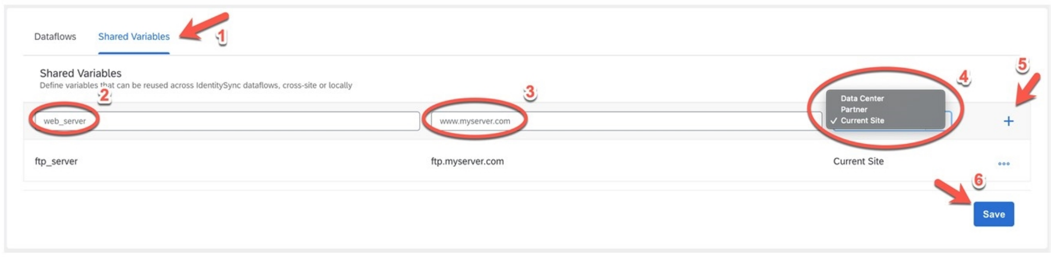
https://developers.gigya.com/display/GD/IdentitySync#IdentitySync-step2

https://developers.gigya.com/display/GD/IdentitySync

For details on editing a workflow: https://developers.gigya.com/display/GD/IdentitySync#IdentitySync-step2

For a code sample of a flow that writes failed records to SFTP, see the Component Repository (https://developers.gigya.com/display/GD/Component+Repository%20#ComponentRepository-Error_Handling_Code_Sample).

This feature allows setting up common settings such as server address, credentials, etc, that can be used on many step configurations. These are the steps for setting up a Shared Variable.



Shared Variables

Define variables that can be reused across IdentitySync dataflows, cross-site or locally

web_server	www.myserver.com	Current Site
ftp_server	ftp.myserver.com	Current Site

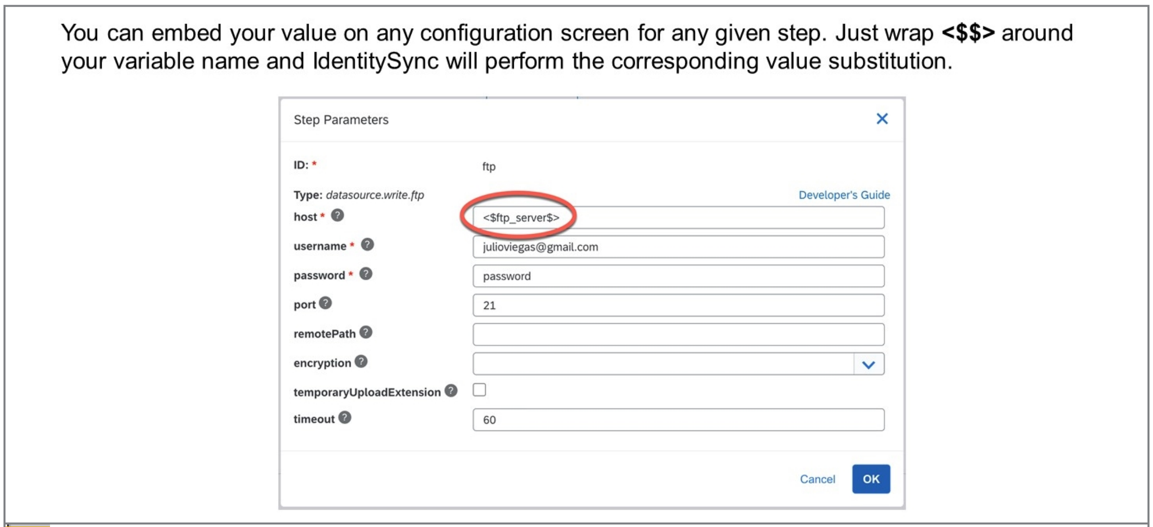
Figure 248: Shared Variables - Setup

Steps for setting up a Shared Variable:

1. Inside Dataflows, click on the tab Shared Variables
2. Name your variable
3. Set a value for your variable
4. Select Scope: This variable applies to all Sites inside a single Data Center, All Data Centers used by a Partner Account i.e. global variable, or only for this particular Site i.e. local variable.

5. Hit the plus icon
6. Hit Save

https://developers.gigya.com/display/GD/IdentitySync#IdentitySync-SharedVariables



You can embed your value on any configuration screen for any given step. Just wrap **<$$>** around your variable name and IdentitySync will perform the corresponding value substitution.

Step Parameters

ID: *	ftp
Type: *datasource.write.ftp*	Developer's Guide
host *	<ftp_server>
username *	julioviegas@gmail.com
password *	password
port	21
remotePath	
encryption	
temporaryUploadExtension	☐
timeout	60

Cancel OK

Figure 249: Shared Variables - Usage

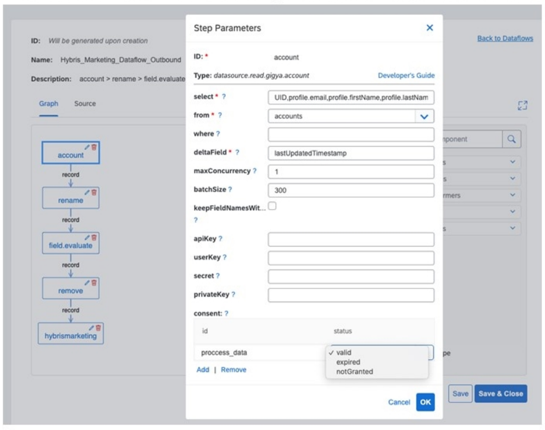
In this case host value <ftp_server> will be replaced by ftp.myserver.com (from the previous slide).

https://en.wikipedia.org/wiki/String_interpolation

Downstreaming Consent Records

Figure 250: Downstreaming Consent Records

Synchronize preferences, consent, and profile data to downstream marketing, sales, and services applications



ID: *Will be generated upon creation*

Name: Hybris_Marketing_Dataflow_Outbound

Description: account > rename > field.evaluate

Step Parameters

ID: * account

Type: *datasource.read.gigya.account*

select * : UID,profile.email,profile.firstName,profile.lastNam

from * : accounts

where :

deltaField * : lastUpdatedTimestamp

maxConcurrency : 1

batchSize : 300

keepFieldNamesWit... : ☐

apiKey :

userKey :

secret :

privateKey :

consent:

id	status
proccess_data	✓ valid / expired / notGranted

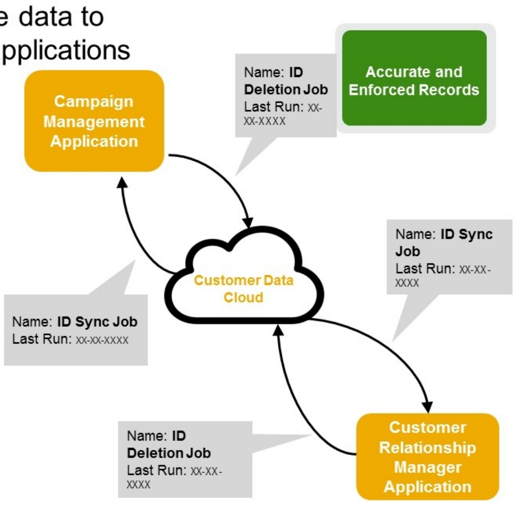


Figure 251: Accurate and Enforced Records

In the screenshot, you can see the configuration of outbound data flow for the downstream application for when an account is removed or renamed

More info: https://developers.gigya.com/display/GD/Mailchimp+Dataflow+-+Outbound

The diagram shows the same cycle of synchronization of profile, consent, and preference data between CDC and the connected downstream applications like CRM and the Campaign Management tool to it.

1. "Enforce consent when syncing data outside of Customer Data Cloud"
2. Making sure only consented data exchanges happen to downstream applications: Marketing Cloud, CRM, DWH, etc.
3. Leverages Identity Sync filtering capabilities and data field level access control

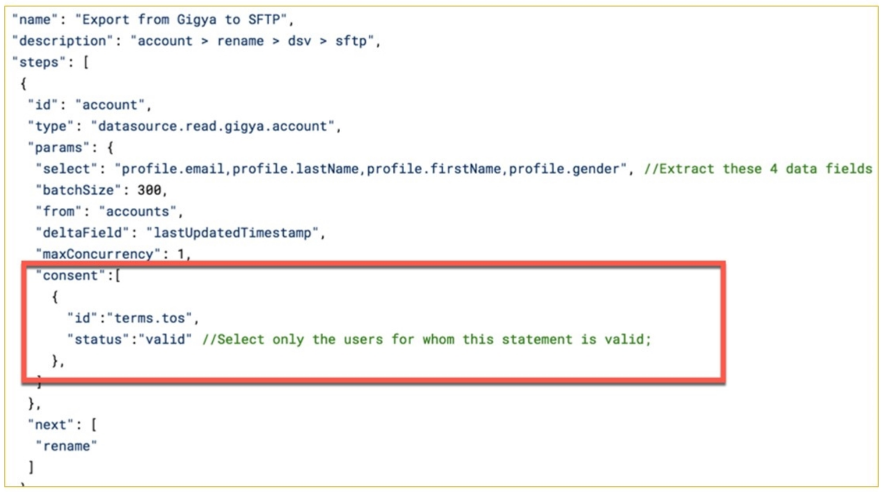
Sync consent-based user data to third-party applications to support the following flows:

- Ensure that only data of users who have a valid consent statement associated with their profile is passed along
- Handle data of users who have withdrawn their consent (e.g. archive, flag for deletion) and those whose consent has expired

Consent enforcement is supported using ***IdentitySync***, Customer Data Cloud's ETL solution

When querying Customer Data Cloud's database using the `datasource.read.gigya.account` component, use the consent parameter to retrieve only users of a given consent status.

Figure 252: Ensuring compliance using Identity Sync



```
"name": "Export from Gigya to SFTP",
"description": "account > rename > dsv > sftp",
"steps": [
 {
  "id": "account",
  "type": "datasource.read.gigya.account",
  "params": {
   "select": "profile.email,profile.lastName,profile.firstName,profile.gender", //Extract these 4 data fields
   "batchSize": 300,
   "from": "accounts",
   "deltaField": "lastUpdatedTimestamp",
   "maxConcurrency": 1,
   "consent":[
     {
       "id":"terms.tos",
       "status":"valid" //Select only the users for whom this statement is valid;
     },
   ]
  },
  "next": [
   "rename"
  ]
```

Figure 253: Sample: export to SFTP users with a valid consent status

Links for reference:

https://developers.gigya.com/display/GD/Export+from+Gigya+to+SFTP

https://developers.gigya.com/display/GD/Consent+Management

Exercise 9

Figure 254: Exercise 9

LESSON SUMMARY

You should now be able to:

- Use SAP Customer Profile, GConnectors, and IdentitySync

UNIT 10 Federation

UNIT OBJECTIVES

- Use SAML, OpenID Connect, and JSON web tokens

Unit 10
Lesson 1

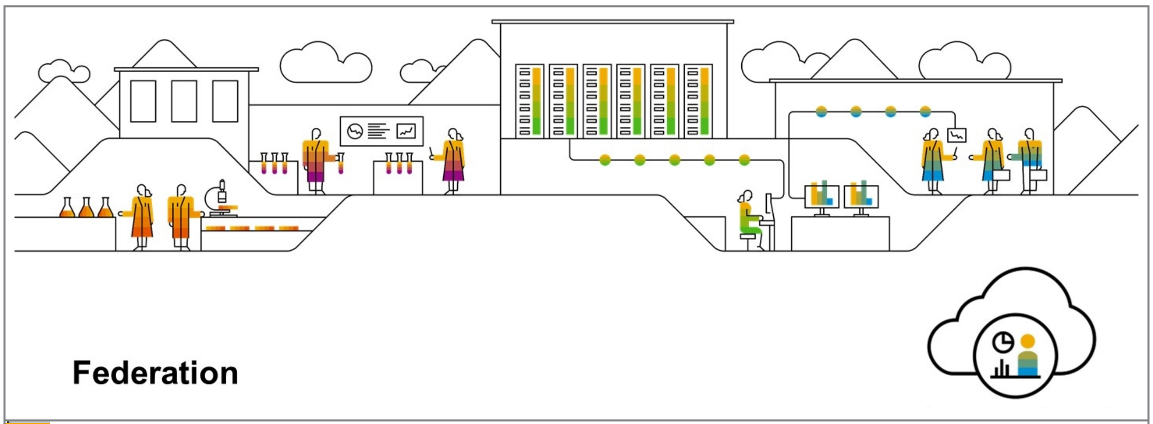
Understanding Federation

LESSON OBJECTIVES

After completing this lesson, you will be able to:

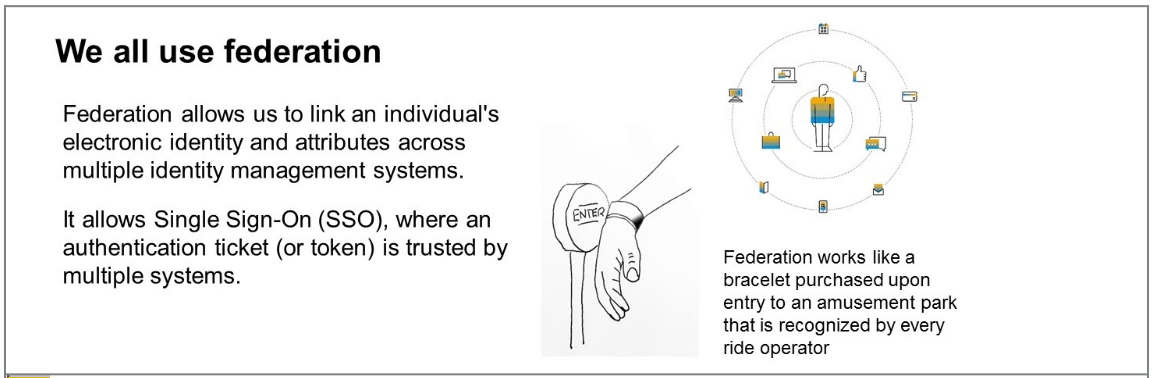
- Use SAML, OpenID Connect, and JSON web tokens

Federation



Federation

Figure 255: Federation



We all use federation

Federation allows us to link an individual's electronic identity and attributes across multiple identity management systems.

It allows Single Sign-On (SSO), where an authentication ticket (or token) is trusted by multiple systems.

Federation works like a bracelet purchased upon entry to an amusement park that is recognized by every ride operator

Figure 256: Federation

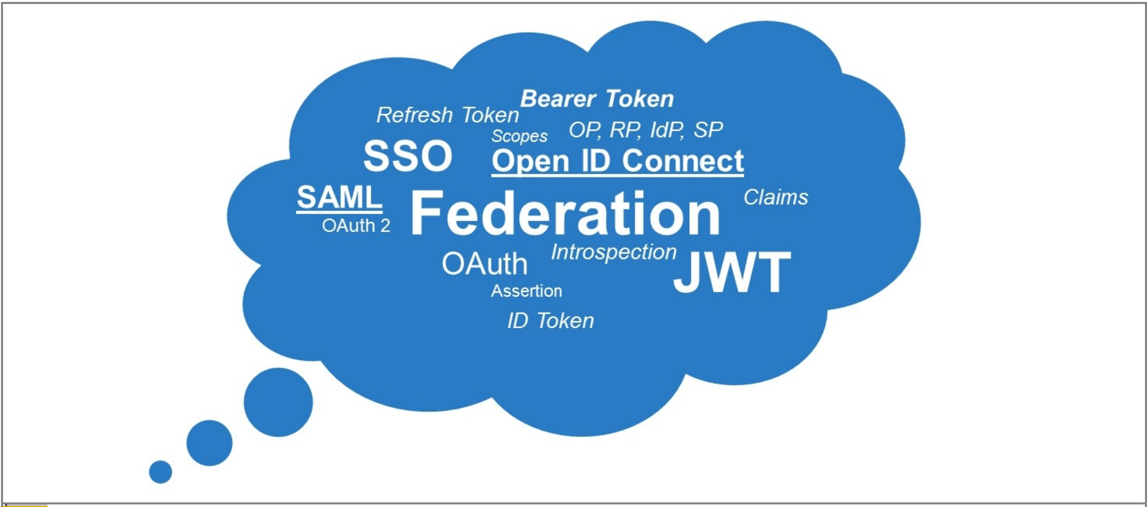
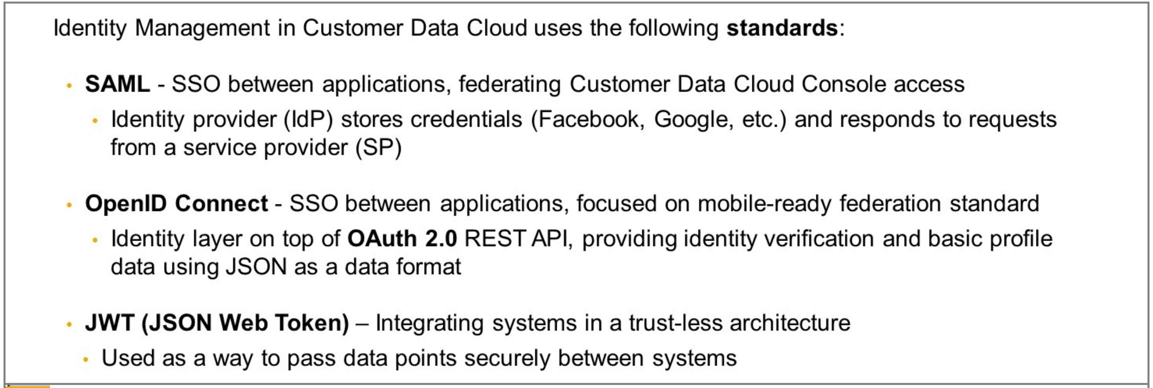


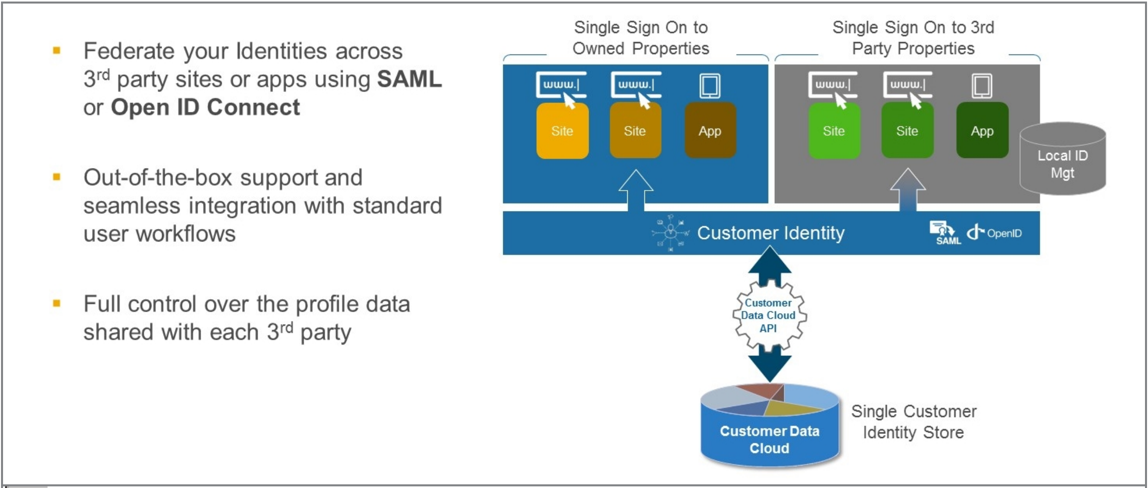
Figure 257: We're Talking Federation When...


Identity Management in Customer Data Cloud uses the following **standards**:

- **SAML** - SSO between applications, federating Customer Data Cloud Console access
  - Identity provider (IdP) stores credentials (Facebook, Google, etc.) and responds to requests from a service provider (SP)
- **OpenID Connect** - SSO between applications, focused on mobile-ready federation standard
  - Identity layer on top of **OAuth 2.0** REST API, providing identity verification and basic profile data using JSON as a data format
- **JWT (JSON Web Token)** – Integrating systems in a trust-less architecture
  - Used as a way to pass data points securely between systems

Figure 258: Standards Support in Customer Data Cloud

SAML - Security Assertion Markup Language


- Federate your Identities across 3rd party sites or apps using **SAML** or **Open ID Connect**
- Out-of-the-box support and seamless integration with standard user workflows
- Full control over the profile data shared with each 3rd party

Single Sign On to Owned Properties
Single Sign On to 3rd Party Properties
www.
Site
Site
App
Local ID Mgt
Customer Identity
SAML
OpenID
Customer Data Cloud API
Customer Data Cloud
Single Customer Identity Store

Figure 259: Federating Your Customer Identities

- Enable authentication via 3rd party Identity Providers using **SAML** or **Open ID Connect**
- Out-of-the-box support and seamless integration with standard user workflows
- Automatically provisions Customer Data Cloud accounts from the 3rd party provider data

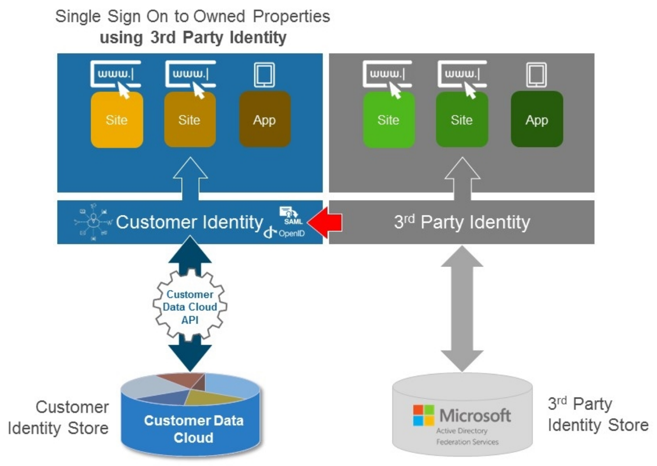


Figure 260: Authentication with External Identity Providers

Security Assertion Markup Language

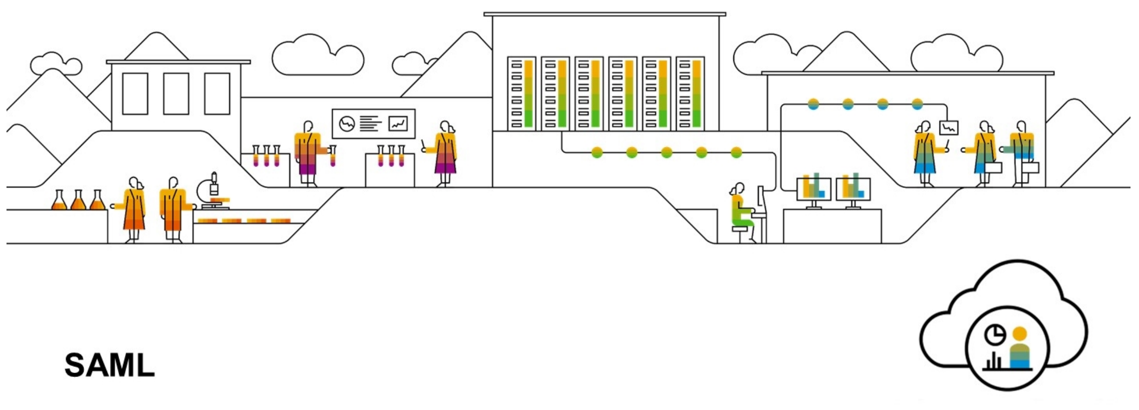


Figure 261: Security Assertion Markup Language

- *Security Assertion Markup Language* (**SAML**) is an XML-based, open-standard data format for exchanging authentication and authorization data between entities. It is also able to exchange user data
- In simple terms, SAML is a standard for federating login/logout to a 3rd party
- There are 3 roles: the p**rincipal** (usually a user), the i**dentity provider** (**IdP**), and the **service provider** (**SP**). The general process is the following:
  1. the principal requests a service from the service provider
  2. the service provider requests and obtains an authentication assertion from the identity provider
  3. on account of this assertion, the service provider can make an access control decision
- SAP Customer Data Cloud can play either the identity provider (IdP) or service provider (SP) in SAML process

Figure 262: Introduction to SAML

SAML Assertions and Flows

An assertion is a package of XML information that provides one or more statements made by a SAML authority. There are 3 types of assertions:

- Authentication assertions are used to make the users prove their identities
- Attribute assertions are used to supply user-specific information. i.e. phone number or email address
- Authorization decisions determine whether the specified subject has been granted or denied permission to access the specified resource

Users can try to access an application from one of the following Login flows:

1. IdP-initiated SSO
2. SP-initiated SSO

Users can try to logout from an application using one of these Logout flows:

1. IdP-initiated SLO
2. SP-initiated SLO

Watch these videos for more info regarding SAML flows (and debugging a SAML connection):

https://enable.cx.sap.com/tag/tagid/saml

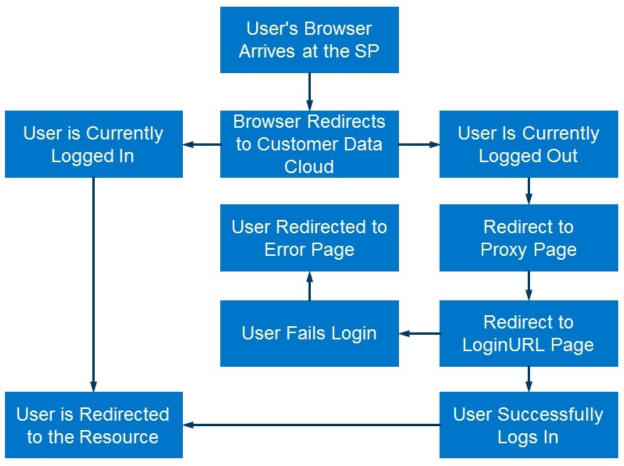
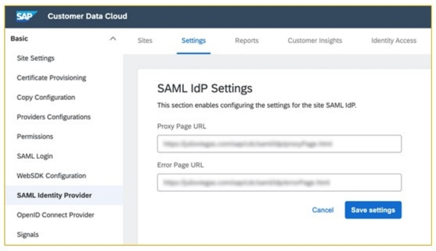

User's Browser Arrives at the SP
User is Currently Logged In
Browser Redirects to Customer Data Cloud
User Is Currently Logged Out
User Redirected to Error Page
Redirect to Proxy Page
User Fails Login
Redirect to LoginURL Page
User is Redirected to the Resource
User Successfully Logs In

Figure 263: Example: SP Initiated SSO (SAP CDC acting as IDP)

For more diagrams on different flows https://developers.gigya.com/display/GD/Gigya+as +SAML+IdP

- Activate your IdP via the CDC admin console *Basic / SAML Identity Provider* page or using the fidm.saml.idp.setConfig API
- Configure SAML IdP settings:
  - **Proxy** page **must** have references to:
    - CDC's SAML SDKs
    - Login URL & Logout URL URIs
  - **Error** page
- Add the configuration for each SP
  - Find out SAML SP Metadata first. If the SP is a CDC site go to ***Basic / SAML Login / SAML SP Metadata***
  - Configure the values for Name, Issuer, Session Lifetime, ACS URL, SLO URL, Binding, Name ID, Attribute Map and x509 Certificates
- Every CDC site is ready to be an IdP, no initial setup needed for issuer, SSO URL, SLO URL and x509 certificate (just check SAML IdP Metadata)

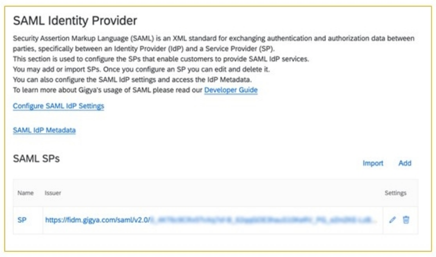


SAML IdP Settings

This section enables configuring the settings for the site SAML IdP.

Proxy Page URL: [redacted]

Error Page URL: [redacted]



SAML Identity Provider

Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between parties, specifically between an Identity Provider (IdP) and a Service Provider (SP).
This section is used to configure the SPs that enable customers to provide SAML IdP services.
You may add or import SPs. Once you configure an SP you can edit and delete it.
You can also configure the SAML IdP settings and access the IdP Metadata.
To learn more about Gigya's usage of SAML please read our Developer Guide

Configure SAML IdP Settings

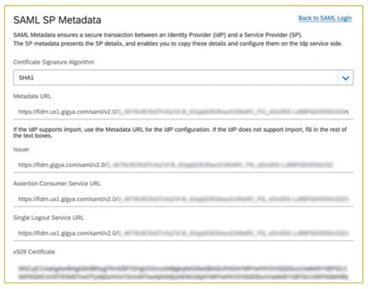
SAML IdP Metadata

SAML SPs

Name	Issuer	Settings
SP	https://fidm.gigya.com/saml/v2.0/[redacted]	

Figure 264: Setup Customer Data Cloud as IdP

- You can register an SP either through your console *Basic / SAML Login* or using the API fidm.saml.setConfig
- Every CDC site is ready to be an SP no initial setup needed for Issuer, ACS URL, SLO URL and x509 certificate (just check SAML SP Metadata)
- Collect the information for your IdP and add the following settings: Name (will be prefixed with saml-), Issuer, SSO URL, Binding, SLO URL, Name ID format, Attribute Map, and x509 Certificate



SAML SP Metadata

SAML Metadata ensures a secure transaction between an Identity Provider (IdP) and a Service Provider (SP).
The SP metadata presents the SP details, and enables you to copy these details and configure them on the Idp service side.

Certificate Signature Algorithm: SHA1

Metadata URL: https://fidm.us1.gigya.com/saml/v2.0/[redacted]/s

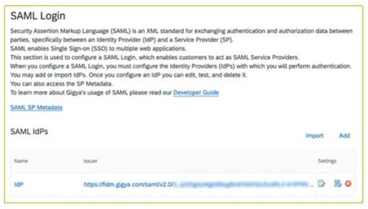
If the IdP supports import, use the Metadata URL for the IdP configuration. If the IdP does not support import, fill in the rest of the text boxes.

Issuer: https://fidm.gigya.com/saml/v2.0/[redacted]

Assertion Consumer Service URL: https://fidm.us1.gigya.com/saml/v2.0/[redacted]

Single Logout Service URL: https://fidm.us1.gigya.com/saml/v2.0/[redacted]

x509 Certificate: [redacted]



SAML Login

Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between parties, specifically between an Identity Provider (IdP) and a Service Provider (SP).
SAML enables Single Sign-on (SSO) to multiple web applications.
This section is used to configure a SAML Login, which enables customers to act as SAML Service Providers.
When you configure a SAML Login, you must configure the Identity Providers (IdPs) with which you will perform authentication.
You may add or import IdPs. Once you configure an IdP you can edit, test, and delete it.
You can also access the SP Metadata.
To learn more about Gigya's usage of SAML please read our Developer Guide

SAML SP Metadata

SAML IdPs

Name	Issuer	Settings
IdP	https://fidm.gigya.com/saml/v2.0/[redacted] ..	

Figure 265: Register a Service Provider (SP)

OpenID Connect

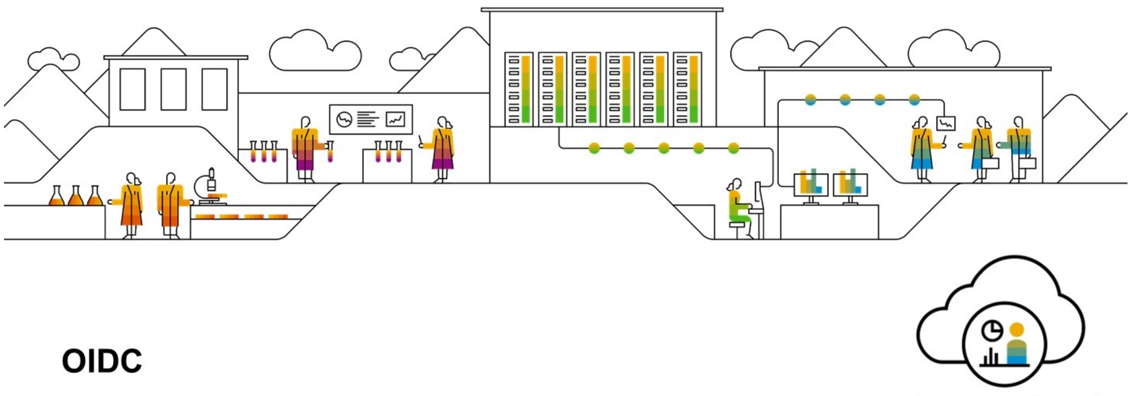
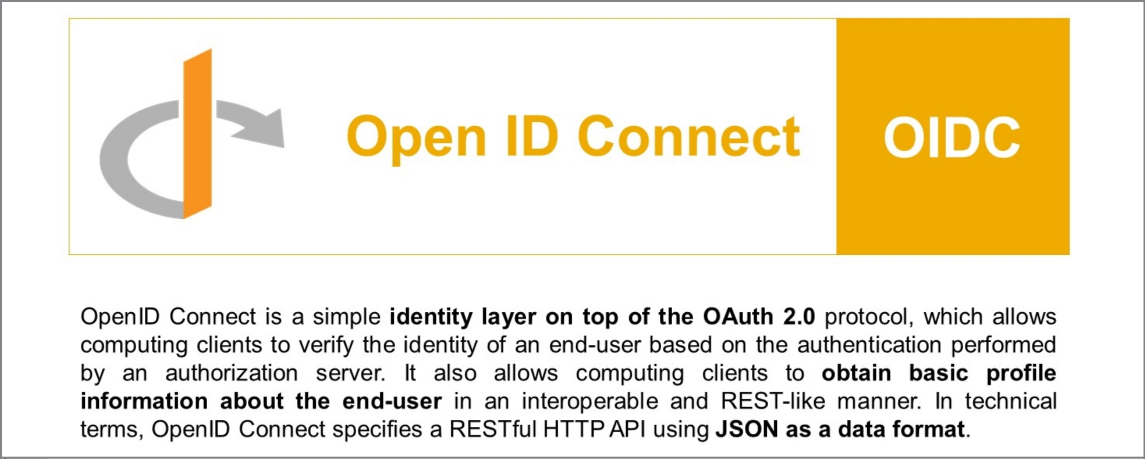


Figure 266: OpenID Connect



Open ID Connect OIDC

OpenID Connect is a simple **identity layer on top of the OAuth 2.0** protocol, which allows computing clients to verify the identity of an end-user based on the authentication performed by an authorization server. It also allows computing clients to **obtain basic profile information about the end-user** in an interoperable and REST-like manner. In technical terms, OpenID Connect specifies a RESTful HTTP API using **JSON as a data format**.

Figure 267: OpenID Connect

CDC supports Auth Code, Implicit, and Hybrid flows using the supported response types: id_token, token, and code. For additional information on the differences in response types see the OpenID.net OIDC specification.

Differences Between OIDC and OAuth 2

- OIDC is an AUTHENTICATION framework; OAuth is AUTHORIZATION
- One of the core advantages of OIDC vs. OAuth 2 is the ID Token
- The ID_Token is a JWT and can be returned from our OIDC implementation

OpenID Connect Provider (OP): An identity provider that is capable of authenticating an end-user and providing claims to a Relying Party. Activating your account as an OP will enable 3rd party sites (relying parties, or RPs) to authenticate their users against your existing user base. For information on setting up your OP configuration, see OpenID Provider Setup.

Relying Party (RP): An OAuth 2.0 client application requiring end user authentication and claiming from an OpenID Provider (OP). For information on setting up your RP configuration, see OpenID Connect RP Setup.

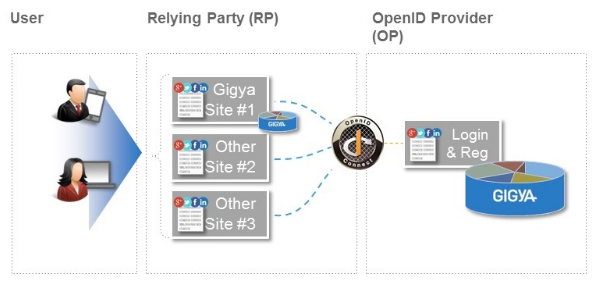
Flow: OpenID provides three separate flows for authenticating users: Authorization Code, Implicit, and Hybrid. For more information, find out more about OpenID Provider Setup and the OpenID specification.

Claim: Information asserted about a user, such as a first name or phone number.

Scope: The type of data to which RPs are granted access. For more information, see allowedScopes.

Tokens:

- Code: Used in the code flow to issue an access token.
- ID Token: A JSON Web Token (JWT) signed by the OP that contains identity information about the user being authenticated. It also contains information about the token itself, such as the time it was issued and its expiration time.
- Access Token: Used to grant or deny access to resources (authorization rather than authentication).
- Refresh Token: Used to generate a new access token



- Native client supported (desktop, mobile, IoT)
- JSON-based rather than XML (SAML)
- Heavy adoption on mobile
- Industry standard SSO solution
- Replacing SAML in many cases

Figure 268: OpenID Connect Benefits

Global Traveller Federated Login

Using Customer Data Cloud, you can act as an OpenID Connect Provider (OP), authenticating users using the OpenID Connect (OIDC) protocol, or as a relying party (RP) that requests user authorization from an OP.

Notes

- Equivalencies from SAML
  - SP -> RP
  - IDP -> OP
- 3 URL's
  - Token
  - Authorize
  - Userinfo

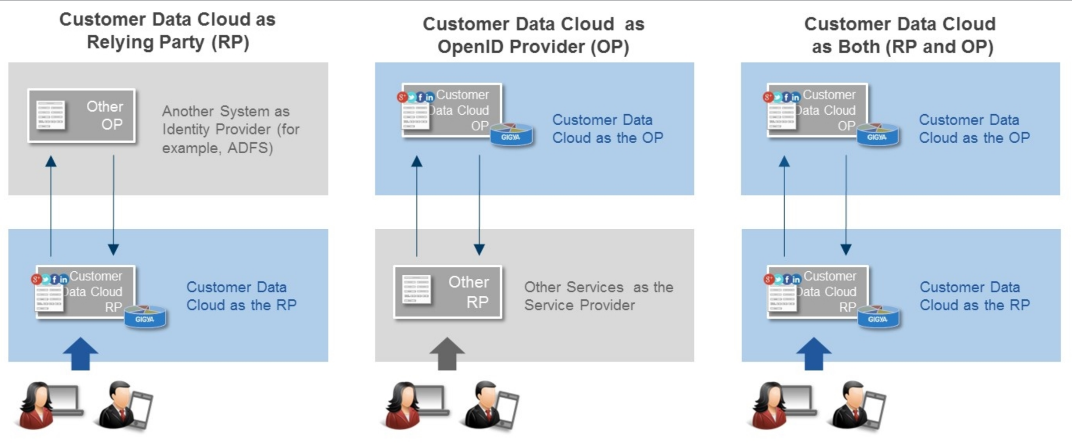
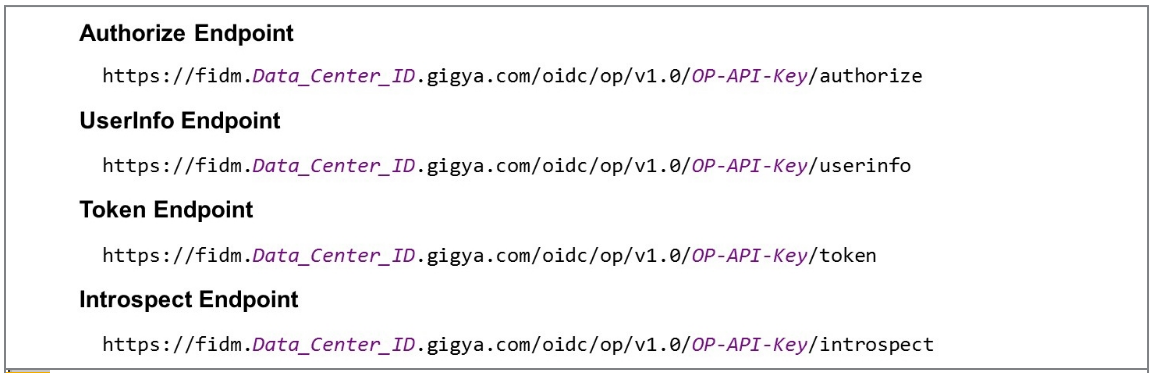

Customer Data Cloud as Relying Party (RP)
Other OP
Another System as Identity Provider (for example, ADFS)
Customer Data Cloud RP
Customer Data Cloud as the RP
Customer Data Cloud as OpenID Provider (OP)
Customer Data Cloud OP
Customer Data Cloud as the OP
Other RP
Other Services as the Service Provider
Customer Data Cloud as Both (RP and OP)
Customer Data Cloud OP
Customer Data Cloud as the OP
Customer Data Cloud RP
Customer Data Cloud as the RP

Figure 269: OIDC Flavors for SAP CDC

Global Traveller Federated Login

Notes

- We can use CDC in 3 different ways
  - RP
    - i.e. CDC screensets and Facebook authentication
  - OP
    - i.e. I'm Ticket Master and CDC identifies the users
  - BOTH
    - i.e. Multiple datacenters...



Authorize Endpoint

`https://fidm.Data_Center_ID.gigya.com/oidc/op/v1.0/OP-API-Key/authorize`

UserInfo Endpoint

`https://fidm.Data_Center_ID.gigya.com/oidc/op/v1.0/OP-API-Key/userinfo`

Token Endpoint

`https://fidm.Data_Center_ID.gigya.com/oidc/op/v1.0/OP-API-Key/token`

Introspect Endpoint

`https://fidm.Data_Center_ID.gigya.com/oidc/op/v1.0/OP-API-Key/introspect`

Figure 270: Customer Data Cloud OIDC Endpoints

Authorization Endpoint: Performs authentication of the user using request parameters defined by OAuth 2.0 and additional parameters and parameter values defined by OpenID Connect. Returns an authorization code. This code should be sent to the token endpoint to receive an id_token and/or access_token.

Token Endpoint: Issues an access_token, id_token, and refresh_token to the RP.

Introspection Endpoint: Used for determining the status of a current access_token (valid or invalid). If the token is valid, it also returns details about the token such as its type, the client_id of the entity that it was issued to, expiration, etc. If the token is invalid, it returns "false", and no additional information.

Token Introspection

Used to determine the current status of a previously received access token. As an RP it may be necessary to check whether an access token you have is still valid.

Claims returned from the Introspect endpoint for a valid access_token are:

- active - boolean (if this is false, no other claims are returned)
- scope
- client_id
- token_type
- exp - expiration time
- iat - issued at
- sub - subject
- aud - audience

- iss - issuer

You construct the request to the introspect endpoint just like you do for the token endpoint, using a Basic Auth token.

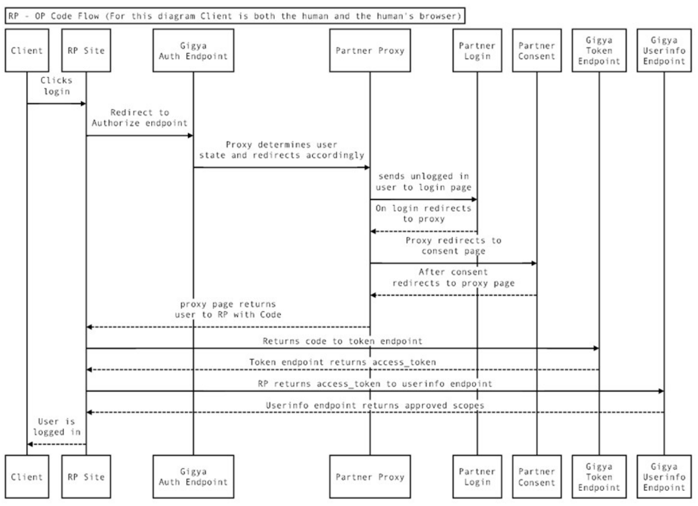

RP - OP Code Flow (For this diagram Client is both the human and the human's browser)
Client
RP Site
Gigya Auth Endpoint
Partner Proxy
Partner Login
Partner Consent
Gigya Token Endpoint
Gigya Userinfo Endpoint
Clicks login
Redirect to Authorize endpoint
Proxy determines user state and redirects accordingly
sends unlogged in user to login page
On login redirects to proxy
Proxy redirects to consent page
After consent redirects to proxy page
proxy page returns user to RP with Code
Returns code to token endpoint
Token endpoint returns access_token
RP returns access_token to userinfo endpoint
Userinfo endpoint returns approved scopes
User is logged in
Client
RP Site
Gigya Auth Endpoint
Partner Proxy
Partner Login
Partner Consent
Gigya Token Endpoint
Gigya Userinfo Endpoint

Figure 271: Detailed Code Flow Diagram with Customer Data Cloud as OP

Detailed CDC flow

This is the flow of the user authentication stage requiring login.

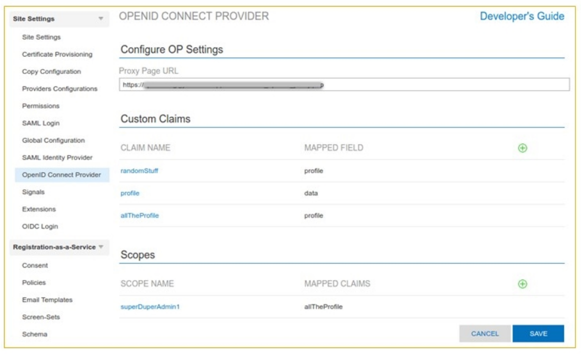
1. The RP generates the request and sends it to the Authorization endpoint.
2. The Authorize endpoint redirects to the OP's proxy page.
3. The Proxy page checks if the user is logged-in.
4. If not logged in, the user is prompted to login and then proceeds to the next step.
5. If logged in, proceeds to next step.
6. The proxy page redirects to the consentURL (Partner Endpoint).
7. If consent is required, the user provides consent. The endpoint then validates the authorization.
8. The consentURL (partner endpoint) redirects back to the proxy page with the requested information (token/code).
9. The proxy page validates the token.
10. The proxy page redirects to the Authorize endpoint.
11. The Authorize endpoint redirects back to the RP.

Notes

- https://developers.gigya.com/display/GD/OpenID+Provider+Setup
- This is the same use case when we use Facebook consent to login....
- A proxy page pointing to 3 pages:
  - Login page
  - Consent page

- Error page
- We need to host these pages outside CDC.

- Activate your OP via console *Basic / OpenID Connect Provider* or using the fidm.oidc.op.setConfig API
- Configure your OP Settings
  - **Setup and activate your proxy page**
    It **must** include:
    - References to CDC's OIDC SDKs.
    - A reference to your Login URL, Consent URL and Error URL URIs.
  - **Create Custom Claims and Scopes**
    By default, CDC supports 3 **standard OIDC scopes**, which are:
    - openid
    - email
    - profile



OPENID CONNECT PROVIDER

Configure OP Settings

Proxy Page URL: https://[redacted]

Custom Claims

CLAIM NAME	MAPPED FIELD
randomStuff	profile
profile	data
allTheProfile	profile

Scopes

SCOPE NAME	MAPPED CLAIMS
superDuperAdmin1	allTheProfile

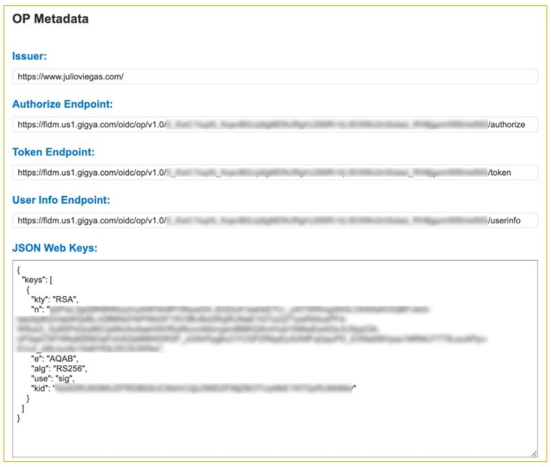
Figure 272: Setup Customer Data Cloud as OP - 1

To add additional custom scopes and/or claims, configure them for your OP here.

Before creating a custom scope, you must define the claims that will be returned within the scope. You can map a custom claim to any available account field from the Custom Claims section of the page.

Once you have defined your claims, you can now define any necessary custom scopes and map these scopes to the required custom claims.

- Every CDC site is ready to be an OP. No initial setup is needed for authorize endpoint, token endpoint, user info endpoint, and the JSON Web Keys (just check OP Metadata)



OP Metadata

Issuer: https://www.julioviegas.com/

Authorize Endpoint: https://fidm.us1.gigya.com/oidc/op/v1.0/[redacted]/authorize

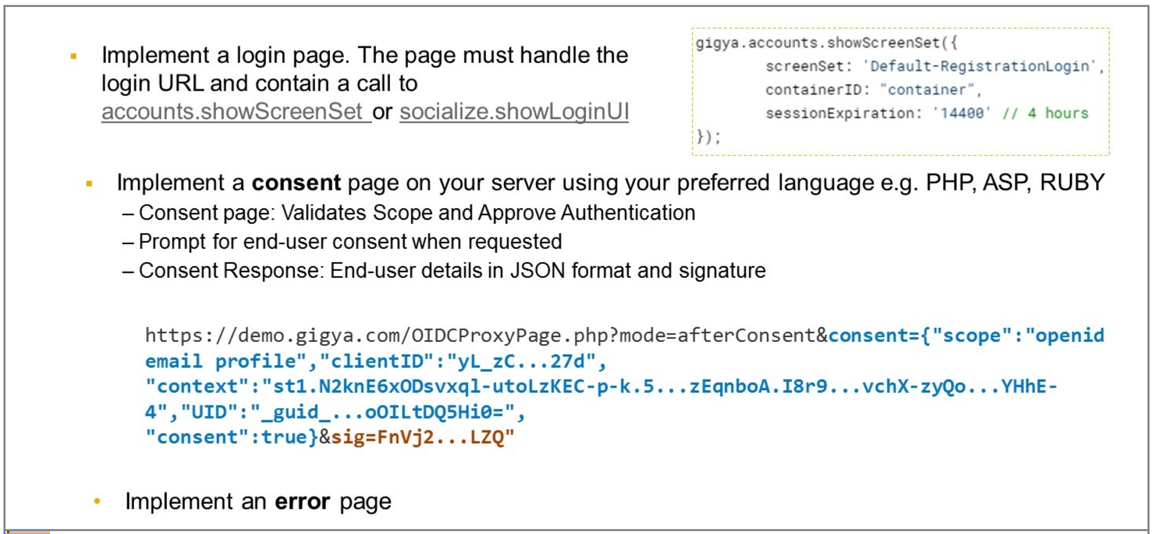
Token Endpoint: https://fidm.us1.gigya.com/oidc/op/v1.0/[redacted]/token

User Info Endpoint: https://fidm.us1.gigya.com/oidc/op/v1.0/[redacted]/userinfo

JSON Web Keys:

```
{
 "keys": [
  {
   "kty": "RSA",
   "n": "[redacted]
   "e": "AQAB",
   "alg": "RS256",
   "use": "sig",
   "kid": "[redacted]"
  }
 ]
}
```

Figure 273: Setup Customer Data Cloud as OP - 2



- Implement a login page. The page must handle the login URL and contain a call to accounts.showScreenSet or socialize.showLoginUI

```
gigya.accounts.showScreenSet({
        screenSet: 'Default-RegistrationLogin',
        containerID: "container",
        sessionExpiration: '14400' // 4 hours
});
```

- Implement a **consent** page on your server using your preferred language e.g. PHP, ASP, RUBY
  – Consent page: Validates Scope and Approve Authentication
  – Prompt for end-user consent when requested
  – Consent Response: End-user details in JSON format and signature

```
https://demo.gigya.com/OIDCProxyPage.php?mode=afterConsent&consent={"scope":"openid
email profile","clientID":"yL_zC...27d",
"context":"st1.N2knE6xODsvxql-utoLzKEC-p-k.5...zEqnboA.I8r9...vchX-zyQo...YHhE-
4","UID":"_guid_...oOILtDQ5Hi0=",
"consent":true}&sig=FnVj2...LZQ"
```

- Implement an **error** page

Figure 274: Setup Customer Data Cloud as OP - 3

More info and code samples here:

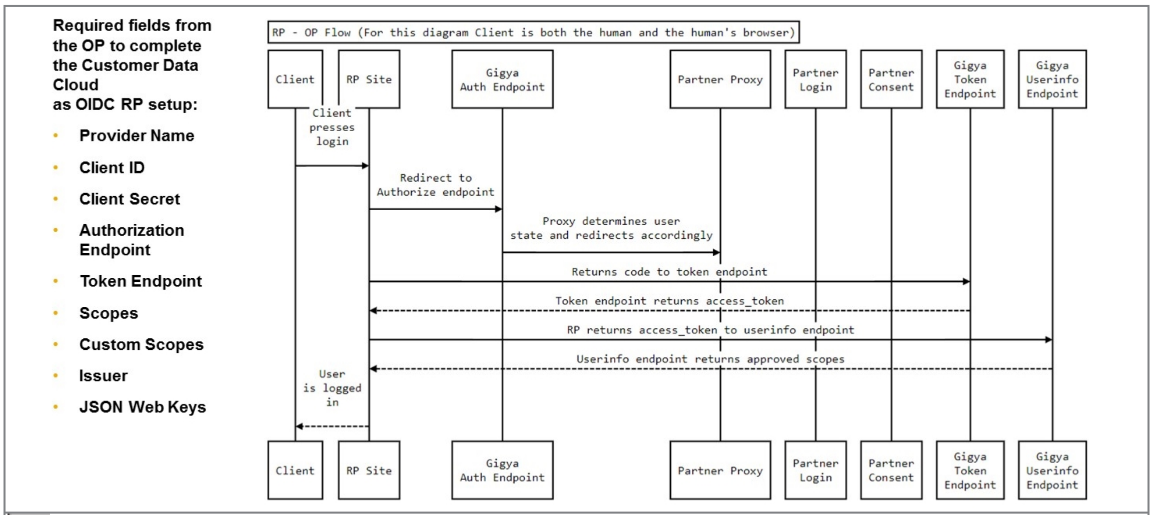
https://developers.gigya.com/display/GD/OpenID+Provider+Setup#OpenIDProviderSetup-ImplementLoginPage

https://developers.gigya.com/display/GD/OpenID+Provider+Setup#OpenIDProviderSetup-ImplementConsentPage

https://developers.gigya.com/display/GD/OpenID+Provider+Setup#OpenIDProviderSetup-ImplementErrorPage

Implement Consent:

1. How to implement Consent Endpoint: https://developers.gigya.com/display/GD/OpenID+Provider+Setup#OpenIDProviderSetup-ConsentEndpoint
2. Consent is handled on your server.
3. Preferred language: PHP, ASP, Ruby, etc
4. Consent page is called from "Proxy Page", once login is successful.
5. Consent page validates the scopes requested by RP. If there is a scope match, Consent page approves the authentication. If there is no scope match, Consent page does not approve the authentication. Consent page may also prompt for the end-user to consent.
6. Consent page then sends the response to Proxy page.
7. The response contains end-user details and signs a consent token.
8. The response parameters, except the signature, must be passed as JSON.
9. The Signature hash is then constructed using that JSON and partner's secret, and the data is returned to the proxy URL page encoded
10. The link in point 1 provides a sample PHP code



Required fields from the OP to complete the Customer Data Cloud as OIDC RP setup:

- **Provider Name**
- **Client ID**
- **Client Secret**
- **Authorization Endpoint**
- **Token Endpoint**
- **Scopes**
- **Custom Scopes**
- **Issuer**
- **JSON Web Keys**

RP - OP Flow (For this diagram Client is both the human and the human's browser)
Client
RP Site
Gigya Auth Endpoint
Partner Proxy
Partner Login
Partner Consent
Gigya Token Endpoint
Gigya Userinfo Endpoint
Client presses login
Redirect to Authorize endpoint
Proxy determines user state and redirects accordingly
Returns code to token endpoint
Token endpoint returns access_token
RP returns access_token to userinfo endpoint
Userinfo endpoint returns approved scopes
User is logged in

Figure 275: Customer Data Cloud as OIDC Relying Party (RP)

For more details visit: https://developers.gigya.com/display/GD/OpenID+Connect+RP+Setup

Visit this page to understand the flow: https://developers.gigya.com/display/GD/OpenID+Connect#OpenIDConnect-DetailedGigyaFlow

Provider Name - The name you will use to reference this OP (must be all alpha-numeric, lowercase characters, and not include any spaces). This value can not be changed once configured. If this provider is ever deleted from your configuration, all users associated with it will be lost.

Client ID - The client_id you received from the OP.

Client Secret - The client-secret you received from the OP.

Authorization Endpoint - The authorize endpoint for the OP.

Token Endpoint - The token endpoint for the OP.

UserInfo Endpoint - The userinfo endpoint for the OP.

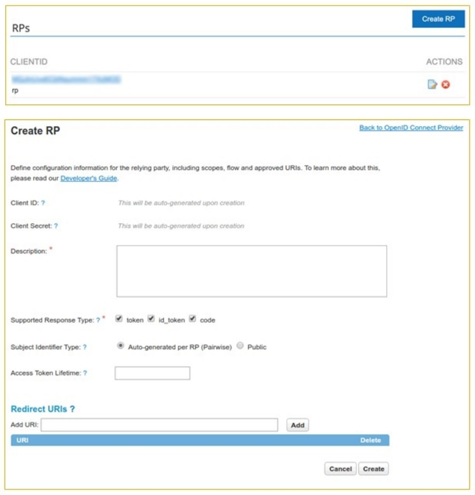
Scopes - Additional standard scopes you are authorized to request from the OP.

Custom Scopes - Any optional custom scopes that you are authorized to request from the OP (a space delimited case-sensitive list of additional scopes that are preconfigured on the OP).

Issuer - The value entered here must match the value returned in the iss claim of the response from the OP. If these values do not match, validation will fail, and users will not be able to login.

JSON Web Keys - The JSON Web Key object containing the keys for the OP. CDC only supports RSA/RS256 keys. If the response can not be validated using the supplied keys, login will fail. If the OP uses any algorithm other than RSA/RS256, you must leave this field blank.

- You can register an RP either through your console *Basic / OIDC Login* or using the API
- The creation and editing of an RP uses the fidm.oidc.op.createRP and the fidm.oidc.op.updateRP APIs.
- You can get the details of an RP via the fidm.oidc.op.getRP API.
- If you are using APIs rather than CDC's console, you should run tests to ensure the RP was configured accurately.



RPs

CLIENTID	ACTIONS
[redacted] rp	

Create RP

Back to OpenID Connect Provider

Define configuration information for the relying party, including scopes, flow and approved URIs. To learn more about this, please read our Developer's Guide.

Client ID: ? This will be auto-generated upon creation

Client Secret: ? This will be auto-generated upon creation

Description: *

Supported Response Type: ? * ✓ token ✓ id_token ✓ code

Subject Identifier Type: ? [✓ Auto-generated per RP (Pairwise)] Public

Access Token Lifetime: ?

Redirect URIs ?

Add URI:

Figure 276: Register a Relying Party (RP)

1. Open the OpenID Connect Provider page on CDC console.
2. Click Configure OP Settings and enter the URL of the proxy page you created earlier.
3. Click Create RP.
4. In the Create RP page, enter a description for this RP.
5. You may limit the RP to only receive specific response types. Under Supported Response Type, select or deselect the types.
6. Under Subject Identifier Type, choose between returning a unique auto-generated ID (pairwise) that is based on the UID, or the user's UID (the Public option).
7. The RP must provide the OP with any redirect URIs they will be using. These URIs must be HTTPS and are entered under Redirect URIs in the respective section. Any requests using redirect URIs that have not been pre-configured will fail.
8. Finally, define an Access Token Lifetime. This is the length of time the granted access_token is valid from the time it was issued and may be from 60 to 604800 seconds (7 days), if left blank, the default is 86400 seconds (24 hours).
9. Click Create.
10. To make changes to the RP, in the list of RPs, click the Edit button:

For more info visit: https://developers.gigya.com/display/GD/OpenID+Provider +Setup#OpenIDProviderSetup-RegisteraRelyingParty(RP)

JSON Web Token

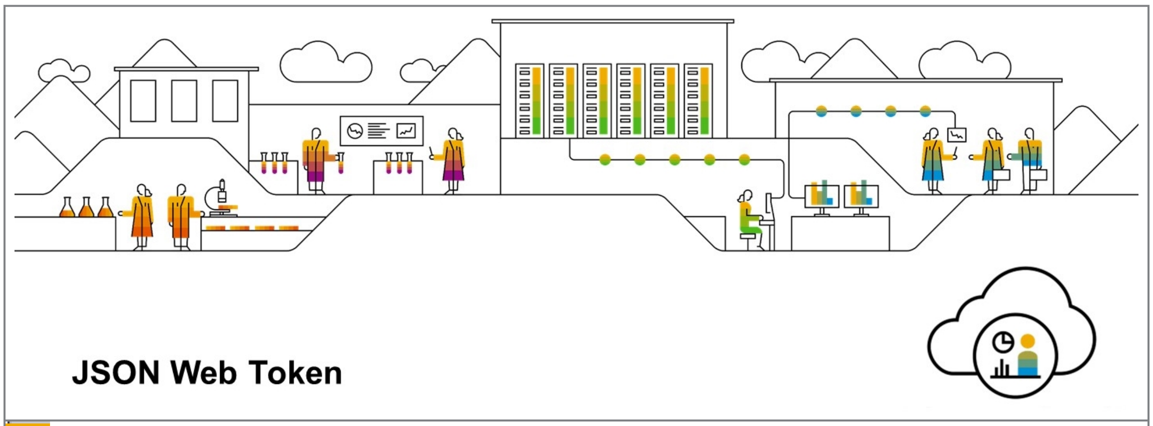
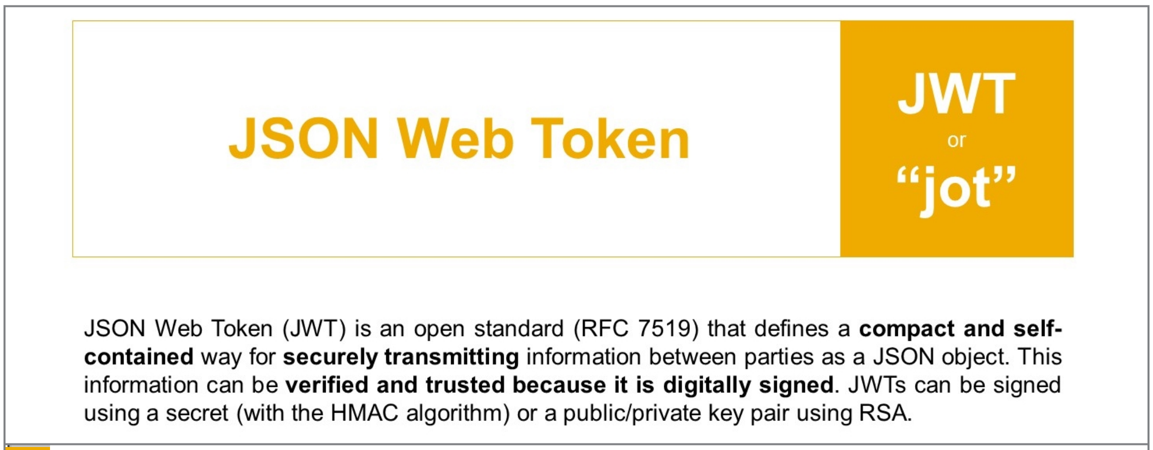


Figure 277: JSON Web Token



JSON Web Token

JWT
or
"jot"

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a **compact and self-contained** way for **securely transmitting** information between parties as a JSON object. This information can be **verified and trusted because it is digitally signed**. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA.

Figure 278: JSON Web Token (JWT)

- https://jwt.io/introduction/
- This is about avoiding man-in-the-middle attacks and tampering
- Customer Data Cloud currently supports generating a JWT on the client SDK and REST API
- Client-side JWT can be requested after login with custom claims (fields and expiration)
- Currently being implemented in the IoT use case and server-to-server trustless environment
- Great option for integrations: it's lightweight, small, and quick
- Google apps, Alexa, and IoT devices are using it.

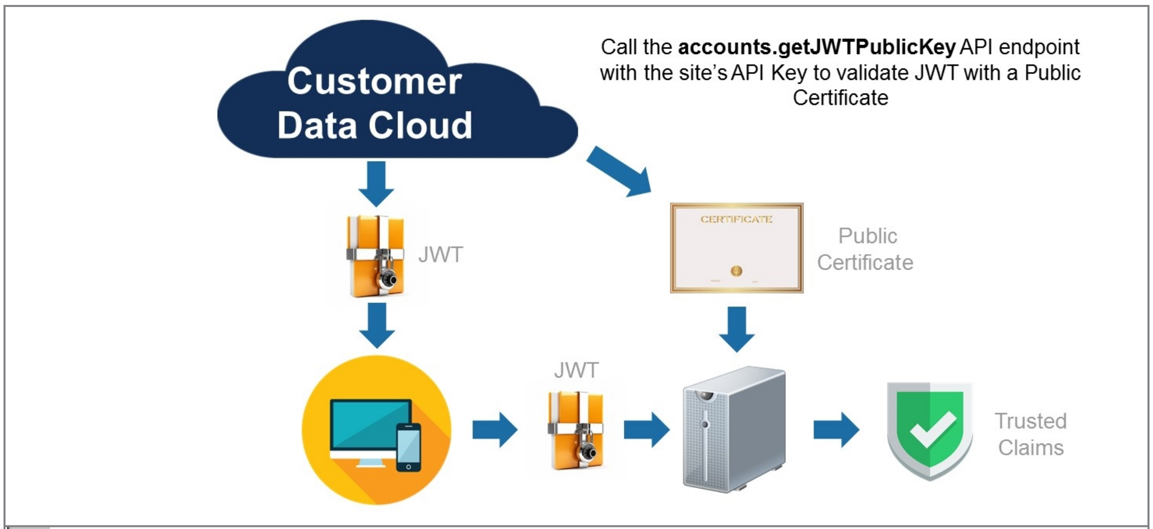
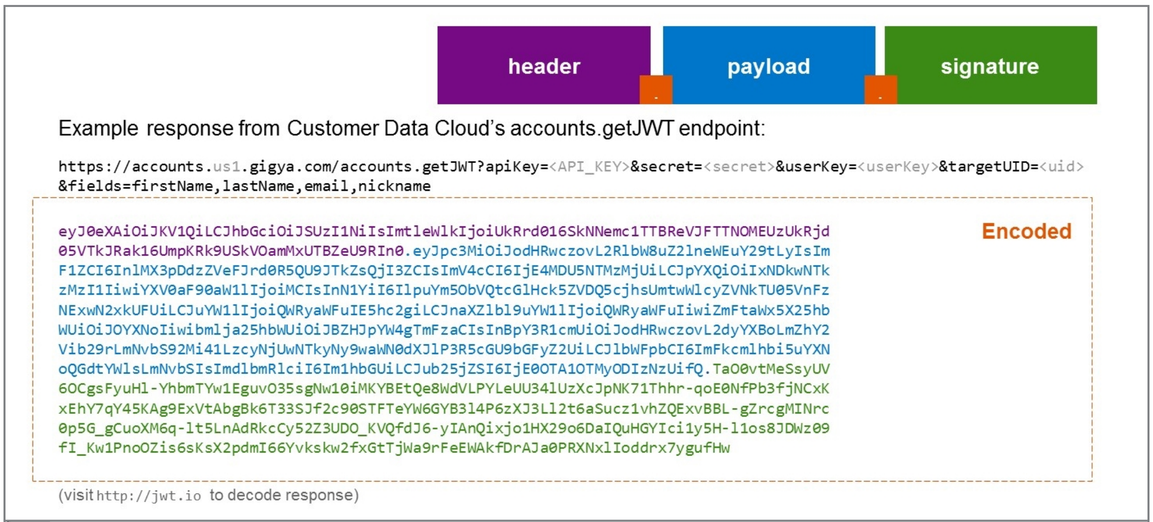

Customer Data Cloud
Call the **accounts.getJWTPublicKey** API endpoint with the site's API Key to validate JWT with a Public Certificate
JWT
CERTIFICATE
Public Certificate
JWT
Trusted Claims

Figure 279: JWT Utilization Example

Imagine a user login to CDC... the response only includes the UID and some other fields. Try to avoid extra CDC calls; that's why we have JWT claims.


header
payload
signature
Encoded

Example response from Customer Data Cloud's accounts.getJWT endpoint:

```
https://accounts.us1.gigya.com/accounts.getJWT?apiKey=<API_KEY>&secret=<secret>&userKey=<userKey>&targetUID=<uid>
&fields=firstName,lastName,email,nickname
```

eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtleWlkIjoiUkRrd016SkNNemc1TTBReVJFTTNOMEUzUkRjd
05VTkJRak16UmpKRk9USkVOamMxUTBZeU9RIn0.eyJpc3MiOiJodHRwczovL2RlbW8uZ2lneWEuY29tLyIsIm
F1ZCI6InlMX3pDdzZVeFJrd0R5QU9JTkZsQjI3ZCIsImV4cCI6IjE4MDU5NTMzMjUiLCJpYXQiOiIxNDkwNTk
zMzI1IiwiYXV0aF90aW1lIjoiMCIsInN1YiI6IlpuYm5ObVQtcGlHck5ZVDQ5cjhsUmtwWlcyZVNkTU05VnFz
NExwN2xkUFUiLCJuYW1lIjoiQWRyaWFuIE5hc2giLCJnaXZlbl9uYW1lIjoiQWRyaWFuIiwiZmFtaWx5X25hb
WUiOiJOYXNoIiwibmlja25hbWUiOiJBZHJpYW4gTmFzaCIsInBpY3R1cmUiOiJodHRwczovL2dyYXBoLmZhY2
Vib29rLmNvbS92Mi41LzcyNjUwNTkyNy9waWN0dXJlP3R5cGU9bGFyZ2UiLCJlbWFpbCI6ImFkcmlhbi5uYXN
oQGdtYWlsLmNvbSIsImdlbmRlciI6Im1hbGUiLCJub25jZSI6IjE0OTA1OTMyODIzNzUifQ.TaO0vtMeSsyUV
6OCgsFyuHl-YhbmTYw1EguvO35sgNw10iMKYBEtQe8WdVLPYLeUU34lUzXcJpNK71Thhr-qoE0NfPb3fjNCxK
xEhY7qY45KAg9ExVtAbgBk6T33SJf2c90STFTeYW6GYB3l4P6zXJ3Ll2t6aSucz1vhZQExvBBL-gZrcgMINrc
0p5G_gCuoXM6q-lt5LnAdRkcCy52Z3UDO_KVQfdJ6-yIAnQixjo1HX29o6DaIQuHGYIci1y5H-l1os8JDWz09
fI_Kw1PnoOZis6sKsX2pdmI66Yvkskw2fxGtTjWa9rFeEWAkfDrAJa0PRXNxlIoddrx7ygufHw

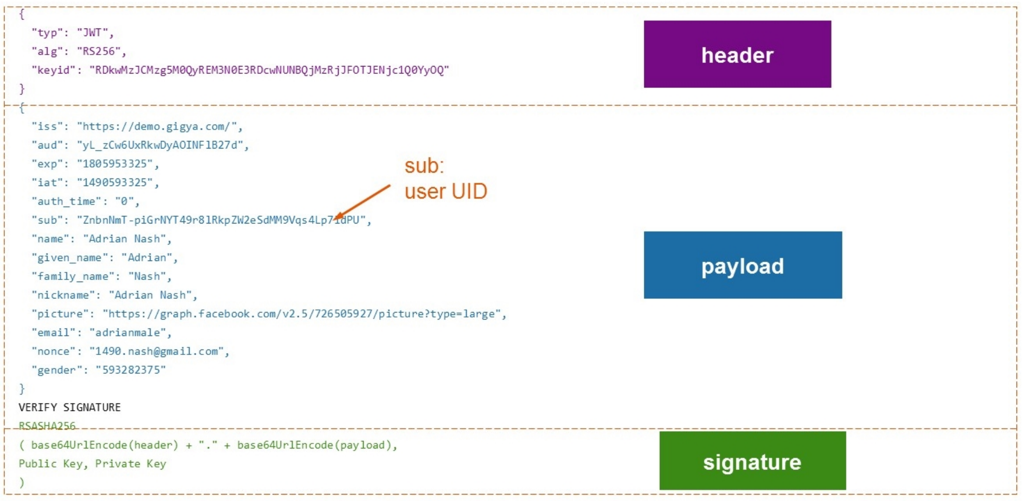
(visit `http://jwt.io` to decode response)

Figure 280: JWT Format

You can go to jwt.io, paste the encoded version, and get the decoded version as a response.

Notes

- This is Base64 encoded



```
{
  "typ": "JWT",
  "alg": "RS256",
  "keyid": "RDkwMzJCMzg5M0QyREM3N0E3RDcwNUNBQjMzRjJFOTJENjc1Q0YyOQ"
}
{
  "iss": "https://demo.gigya.com/",
  "aud": "yL_zCw6UxRkwDyAOINF1B27d",
  "exp": "1805953325",
  "iat": "1490593325",
  "auth_time": "0",
  "sub": "ZnbnNmT-piGrNYT49r81RkpZW2eSdMM9Vqs4Lp71dPU",
  "name": "Adrian Nash",
  "given_name": "Adrian",
  "family_name": "Nash",
  "nickname": "Adrian Nash",
  "picture": "https://graph.facebook.com/v2.5/726505927/picture?type=large",
  "email": "adrianmale",
  "nonce": "1490.nash@gmail.com",
  "gender": "593282375"
}
VERIFY SIGNATURE
RSASHA256
( base64UrlEncode(header) + "." + base64UrlEncode(payload),
Public Key, Private Key
)
```

Figure 281: JWT Format Decoded

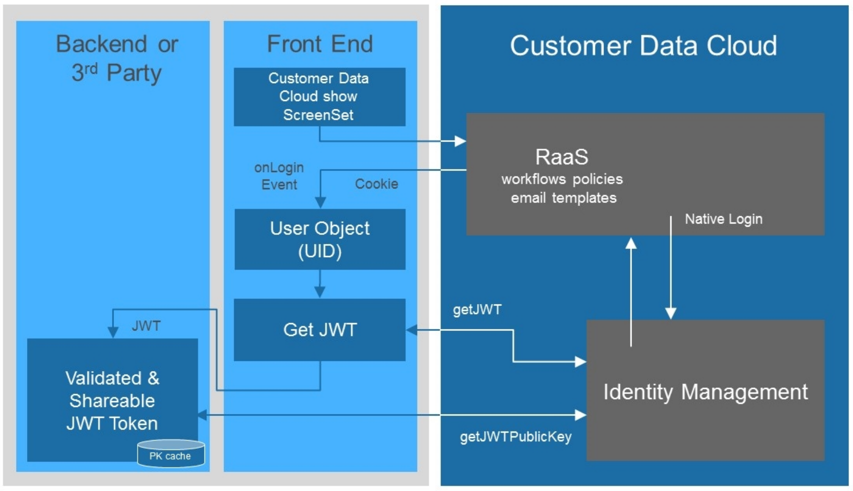


Figure 282: JWT Usage - Front End option

Use Case: we want access to a JWT, specifically to its payload to retrieve some trusted content (retrieve several user info fields) for the currently logged in user ONLY from a backend or third-party system.

Flow Steps:

1. SAP CDC presents the login screenset and the user logs in into CDC Identity Manager using the current RaaS policies and setup.

2. Then the onLogin event is triggered on the client-side and the User Object with the UID field is returned. This information also is stored into a cookie: https://developers.gigya.com/display/GD/Advanced+Cookie+Reference

3. Once the user is authenticated, the user can use the Web SDK (Client API) to retrieve a JWT (JSON Web Token) from the Identity Manager as is explained in next slide.

https://developers.gigya.com/display/GD/accounts.getJWT+JS

4. The JWT must be validated using the public key obtained from the server via the REST API method (notice that we are using a different API!)

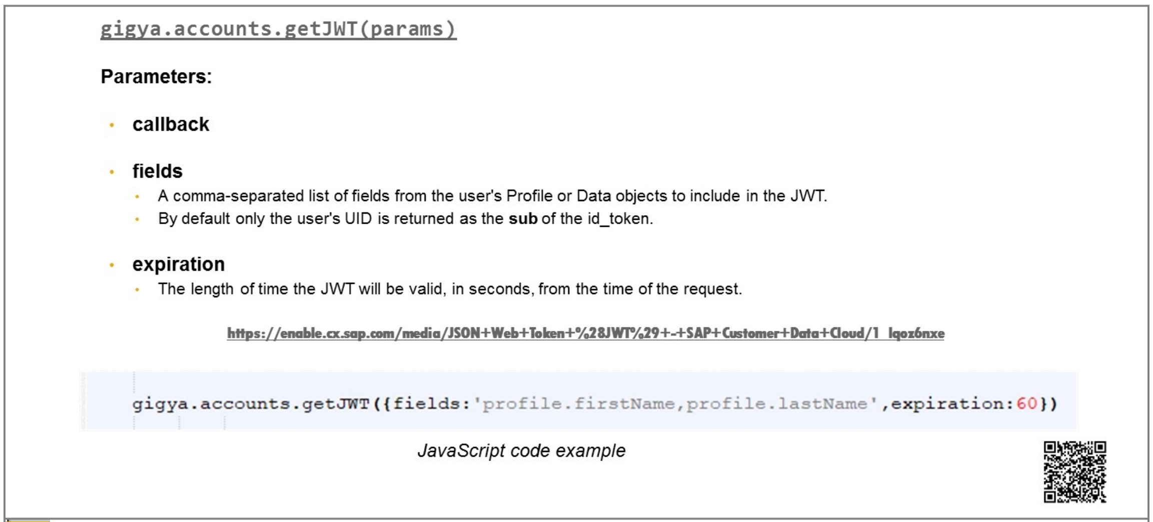
https://developers.gigya.com/display/GD/accounts.getJWTPublicKey+REST

These public keys are usually cached to avoid round trips, although from time-to-time they changed, so we need to check if the cached version is still valid using the keyid field.

https://knowledge.gigya.com/GKB_Article?id=a6H400000004DZpEAM&source=a1Y1W000008BuBrUAK

Sample business scenarios/use cases:

- Front end usage: Client-side login (i.e. email/pwd with SAP CDC screenSets, social login) is a flow that natively occurs on a user's device (browser/mobile). In the case of social login, it also requires the user to successfully login to one of the providers supported by CDC. Obviously, this process has to happen on the client side, which means there is a risk that a malicious user will try to tamper with the data sent from the client to the server. The most important piece of data is the UID, which is used to authenticate the user and log him/her into your system. Once the user completes log in (front-end), you would probably want to send the UID to the server for further processing (adding user to a database, create a server-side session, etc). During this process, you want to prevent tampering with the UID. Instead of just sending a UID to the server, a client-side JWT can be generated, then sent to your server. The JWT includes a signature that is used to verify the message wasn't changed along the way. This is the most common scenario for using JWT.
- Other use cases: Authorize user against protected resources using JWT profile/custom data fields. For instance, a client-side generated JWT containing a membership type field (data.membershipType) can be used to request a resource that will be available to premium users only (i.e. data.membershipType == "premium", then... return video content, else... access denied). Share profile information. A client-side generated JWT can be used as a simple mechanism to transmit/exchange profile data. For instance, in order to submit a claim, a user must fill out a form on an external website and a JWT could be sent as part of the "redirection" request. The target website can retrieve profile information (first name, last name, email) from the JWT and auto-populate the form (or even authenticate the user if exists).



gigya.accounts.getJWT(params)

Parameters:

- **callback**
- **fields**
  - A comma-separated list of fields from the user's Profile or Data objects to include in the JWT.
  - By default only the user's UID is returned as the **sub** of the id_token.
- **expiration**
  - The length of time the JWT will be valid, in seconds, from the time of the request.

https://enable.cx.sap.com/media/JSON+Web+Token+%28JWT%29+-+SAP+Customer+Data+Cloud/1_lqoz6nxe

```
gigya.accounts.getJWT({fields:'profile.firstName,profile.lastName',expiration:60})
```

JavaScript code example

Figure 283: Client Side JWT API Call example - Web SDK

More info about parameters: https://developers.gigya.com/display/GD/accounts.getJWT+JS

https://enable.cx.sap.com/media/JSON+Web+Token+%28JWT%29+-+SAP+Customer+Data+Cloud/1_lqoz6nxe

gigya.accounts.getJWT({fields:'profile.firstName,profile.lastName',expiration:60})

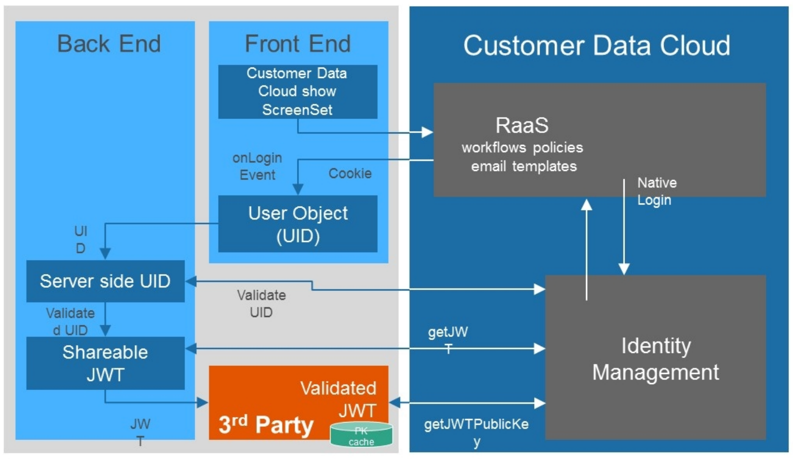


Figure 284: JWT Usage - Server-Side

Use Case: we want access to a JWT, specifically to its payload to retrieve some trusted content (several user info fields) for ANY logged in user from a backend or third-party system.

Flow Steps:

1. SAP CDC presents the login screenset and the user logs in into CDC Identity Manager using the current RaaS policies and setup.

2. Then, the onLogin event is triggered on the client side and the User Object with the UID field is returned. This information is also stored in a cookie: https://developers.gigya.com/display/GD/Advanced+Cookie+Reference

These first two steps are equal to the Front-end option, but from this point on the flow is different.

3. For any user authenticated (logged in), we can use the Rest API (or the server-side SDK) to validate any UID for any logged user. This can be done using SigUtils.validateUserSignature() in any server SDK or by checking UID, signatureTimestamp, and UIDSignature fields using the Signature validation process.

https://developers.gigya.com/display/GD/accounts.getJWT+REST

https://developers.gigya.com/display/GD/User+JS (User Object and its fields)

https://developers.gigya.com/display/GD/Security+Best+Practices#SecurityBestPractices-SignatureValidationProcess (how to check if a signature is valid)

4. After the UID is validated on the server-side, we can use the Rest API (or any server side SDK) to retrieve a JWT (JSON Web Token) from the Identity Manager as is explained in next slide.

5. The JWT must be validated using the public key obtained from the server via the REST API method and accessible for the back-end application.

https://developers.gigya.com/display/GD/accounts.getJWTPublicKey+REST

These public keys are usually cached to avoid round trips, although from time-to-time they changed, so we need to check if the cached version is still valid using the keyid field

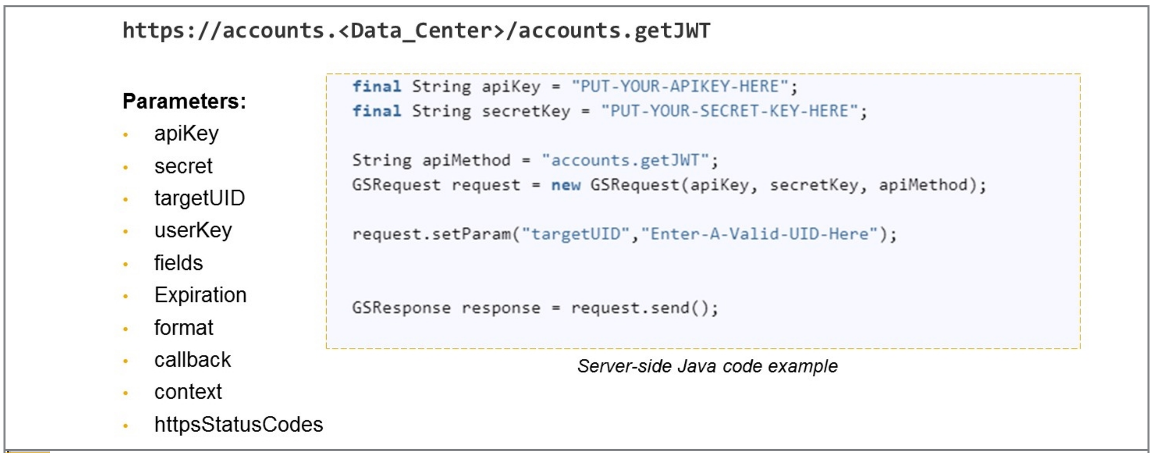
https://knowledge.gigya.com/GKB_Article?id=a6H400000004DZpEAM&source=a1Y1W000008BuBrUAK

In this option you don't have to pass the UID to the backend/3rd Party

Sample business scenarios/use cases:

1. Server-side usage: Although less common than client-side generated JWT, there are some use cases for server-side usage. For instance server-to-server federation. Also for instance, a user (logged in w/ server-side session) in an e-Commerce portal must be redirected to a partner payment system. Before redirection, the e-Commerce back-end can request a user's JWT to SAP CDC via a REST API call, just by passing a UID in a signed request. Once JWT is retrieved, it can be transmitted to the partner payment system for further JWT validation and authentication on the payment system.

2. User impersonation: For example, considering a user (user 1) with authorization to approve leave requests on behalf of someone else (user 2), the system (server-side) could ask SAP CDC to generate at any time a JWT for any user (i.e. user 2). Then, this JWT could be used as an impersonation mechanism to allow user 1 to approve tasks on another system on behalf of user 2. IoT and device registration. The objective is to associate an identity with a device (car, home assistance, smartwatch, other), by allowing the users to login/register on a secondary device e.g. a mobile phone, tablet or laptop that is easier to type on. This can be done by using either a pin code or a QR code, generating a JWT (server-side) linked to the user. The JWT is retrieved by the device (once the pin code is validated), sent with subsequent requests, and acts as an access token to provide direct access to accounts for reads / writes.



```
https://accounts.<Data_Center>/accounts.getJWT
```

Parameters:

- apiKey
- secret
- targetUID
- userKey
- fields
- Expiration
- format
- callback
- context
- httpsStatusCodes

```
final String apiKey = "PUT-YOUR-APIKEY-HERE";
final String secretKey = "PUT-YOUR-SECRET-KEY-HERE";

String apiMethod = "accounts.getJWT";
GSRequest request = new GSRequest(apiKey, secretKey, apiMethod);

request.setParam("targetUID","Enter-A-Valid-UID-Here");


GSResponse response = request.send();
```

Server-side Java code example

Figure 285: Server-Side JWT API Call example - REST API

For More info on parameters, see https://developers.gigya.com/display/GD/accounts.getJWT+REST#accounts.getJWTREST-Parameters

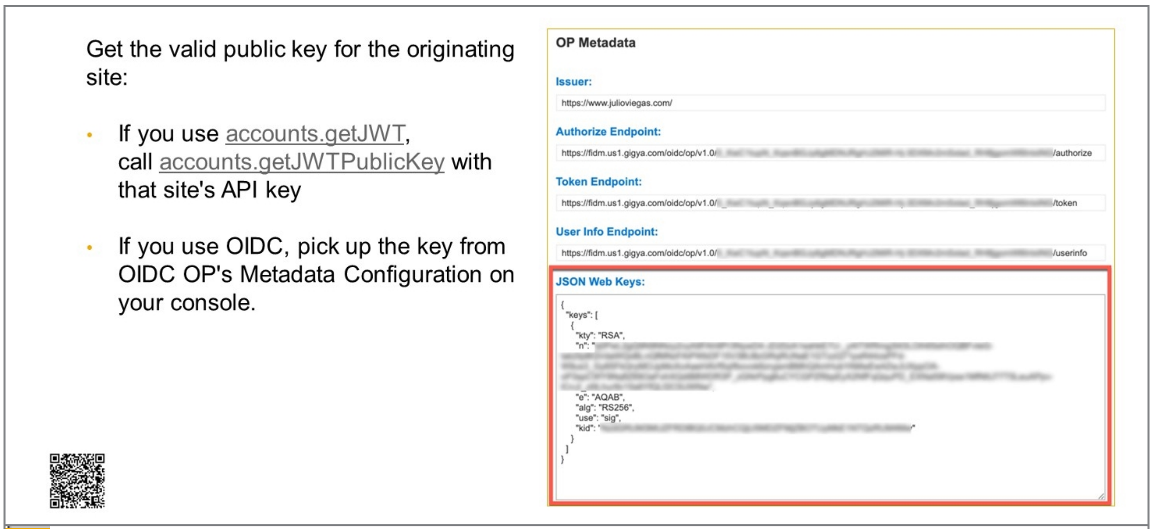
targetUID (string): The UID of the user whose data is being requested. Must be a user for the site of the associated apiKey (above).

Fields (string, comma-delimited): Any existing profile and/or data fields in the target site's database you want to explicitly return in the JWT for this targetUID.

- When requesting profile fields, it is not necessary to prepend 'profile.' (e.g., profile.firstName can be passed as firstName).

Expiration (integer) - The TTL of the returned JWT, in seconds. If this parameter is not passed, the default is 300.

format and callback - used when you want to specify a callback method to be called with the response (set format to 'jsonp' and callback to method name)



Get the valid public key for the originating site:

- If you use accounts.getJWT, call accounts.getJWTPublicKey with that site's API key
- If you use OIDC, pick up the key from OIDC OP's Metadata Configuration on your console.

OP Metadata

Issuer:
https://www.julioviegas.com/

Authorize Endpoint:
https://fidm.us1.gigya.com/oidc/op/v1.0/[redacted]/authorize

Token Endpoint:
https://fidm.us1.gigya.com/oidc/op/v1.0/[redacted]/token

User Info Endpoint:
https://fidm.us1.gigya.com/oidc/op/v1.0/[redacted]/userinfo

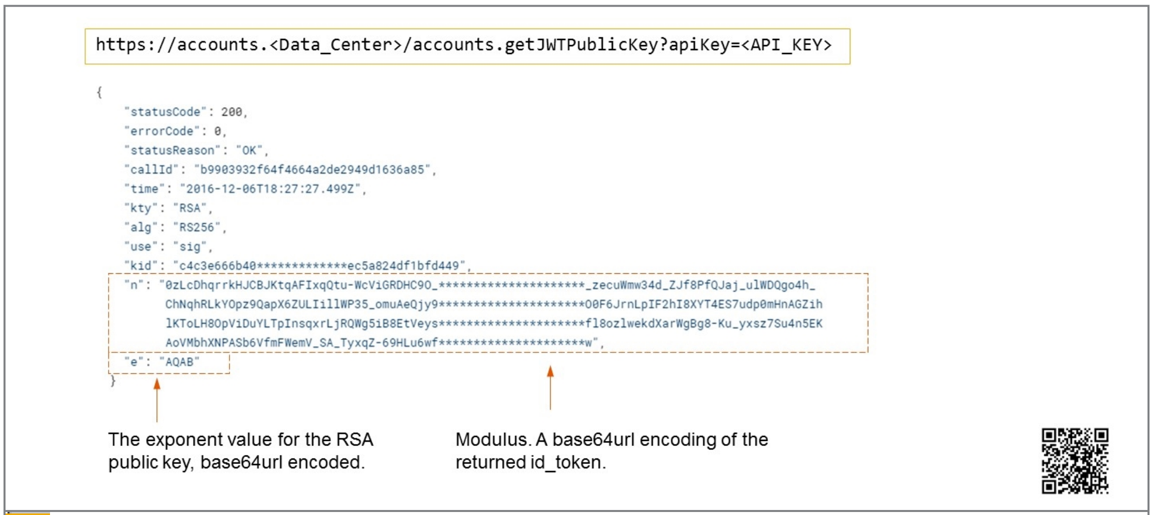
JSON Web Keys:

```
{
  "keys": [
    {
      "kty": "RSA",
      "n": "[redacted]
      "e": "AQAB",
      "alg": "RS256",
      "use": "sig",
      "kid": "[redacted]"
    }
  ]
}
```

Figure 286: Public key for validating a JWT

This is a screenshot from CDC console > Settings > OpenID Connect Provider > OP Metadata (link on page).

More info here: https://developers.gigya.com/display/GD/How+To+Validate+A+Gigya +id_token#HowToValidateAGigyaid_token-Validation



```
https://accounts.<Data_Center>/accounts.getJWTPublicKey?apiKey=<API_KEY>
```

```
{
    "statusCode": 200,
    "errorCode": 0,
    "statusReason": "OK",
    "callId": "b9903932f64f4664a2de2949d1636a85",
    "time": "2016-12-06T18:27:27.499Z",
    "kty": "RSA",
    "alg": "RS256",
    "use": "sig",
    "kid": "c4c3e666b40************ec5a824df1bfd449",
    "n": "0zLcDhqrrkHJCBJKtqAFIxqQtu-WcViGRDHC90_*********************_zecuWmw34d_ZJf8PfQJaj_ulWDQgo4h_
          ChNqhRLkYOpz9QapX6ZULIillWP35_omuAeQjy9*********************O0F6JrnLpIF2hI8XYT4ES7udp0mHnAGZih
          lKToLH8OpViDuYLTpInsqxrLjRQWg5iB8EtVeys*********************fl8ozlwekdXarWgBg8-Ku_yxsz7Su4n5EK
          AoVMbhXNPASb6VfmFWemV_SA_TyxqZ-69HLu6wf*********************w",
    "e": "AQAB"
}
```

The exponent value for the RSA public key, base64url encoded.

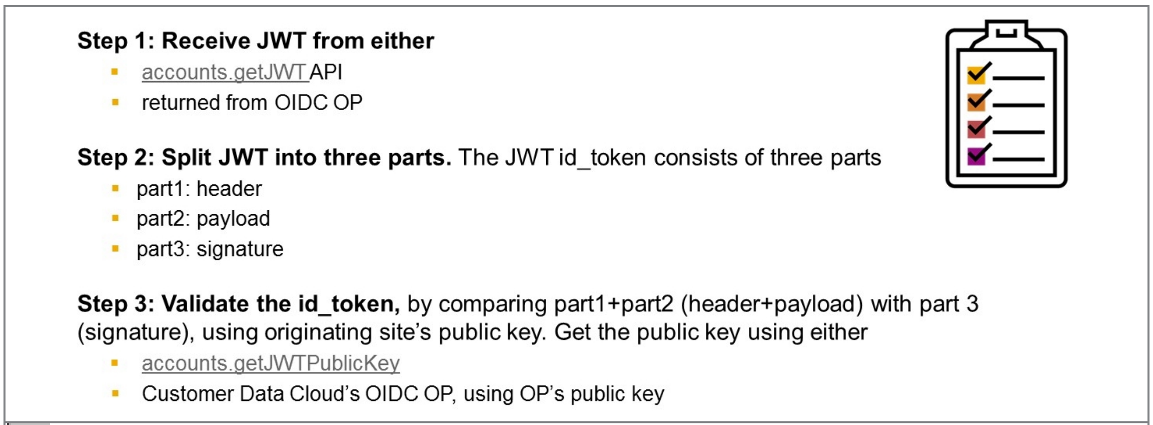
Modulus. A base64url encoding of the returned id_token.

Figure 287: Example accounts.getJWTPublicKey Response

https://developers.gigya.com/display/GD/accounts.getJWTPublicKey+REST

Reminder: <Data_Center> can be:

- us1.gigya.com - For the US data center.
- eu1.gigya.com - For the European data center.
- au1.gigya.com - For the Australian data center.
- ru1.gigya.com - For the Russian data center.
- cn1.gigya-api.cn - For the Chinese data center.



Step 1: Receive JWT from either

- accounts.getJWT API
- returned from OIDC OP

Step 2: Split JWT into three parts. The JWT id_token consists of three parts

- part1: header
- part2: payload
- part3: signature

Step 3: Validate the id_token, by comparing part1+part2 (header+payload) with part 3 (signature), using originating site's public key. Get the public key using either

- accounts.getJWTPublicKey
- Customer Data Cloud's OIDC OP, using OP's public key

Figure 288: Steps to validate a Customer Data Cloud JWT id_token

Refer the following page for sample code

https://developers.gigya.com/display/GD/How+To+Validate+A+Gigya +id_token#HowToValidateAGigyaid_token-Steps

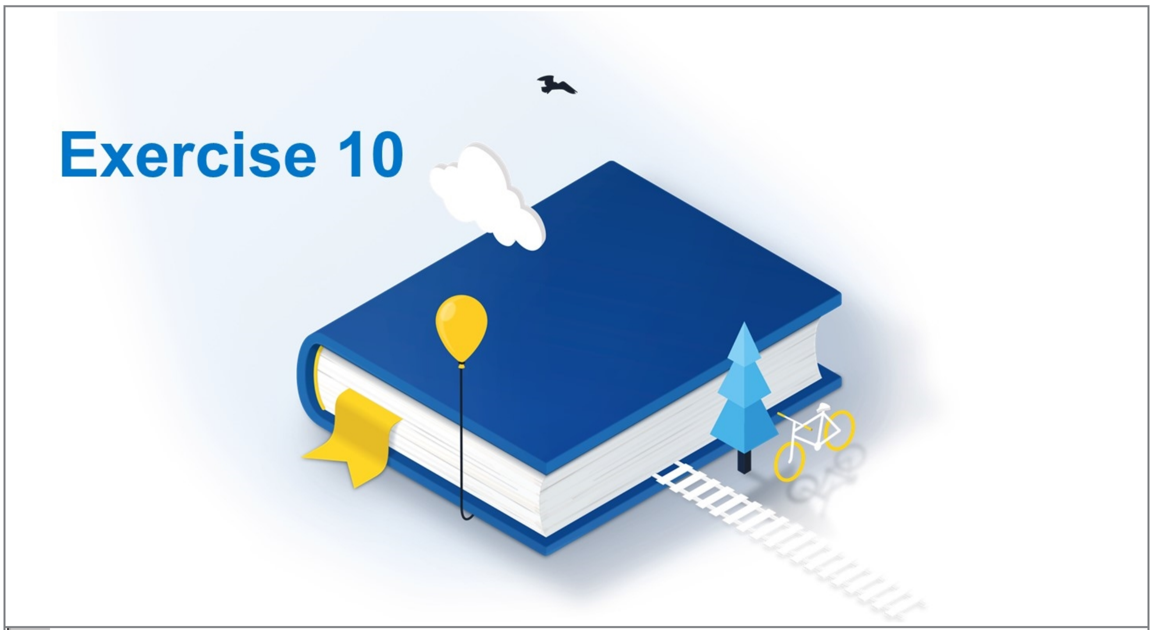


Figure 289: Exercise 10

LESSON SUMMARY

You should now be able to:

- Use SAML, OpenID Connect, and JSON web tokens

UNIT 11 Mobile Apps Integration

UNIT OBJECTIVES

- Use mobile SDKs and mobile session management

Unit 11
Lesson 1

Understanding Mobile Apps Integration

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use mobile SDKs and mobile session management

Mobile SDKs

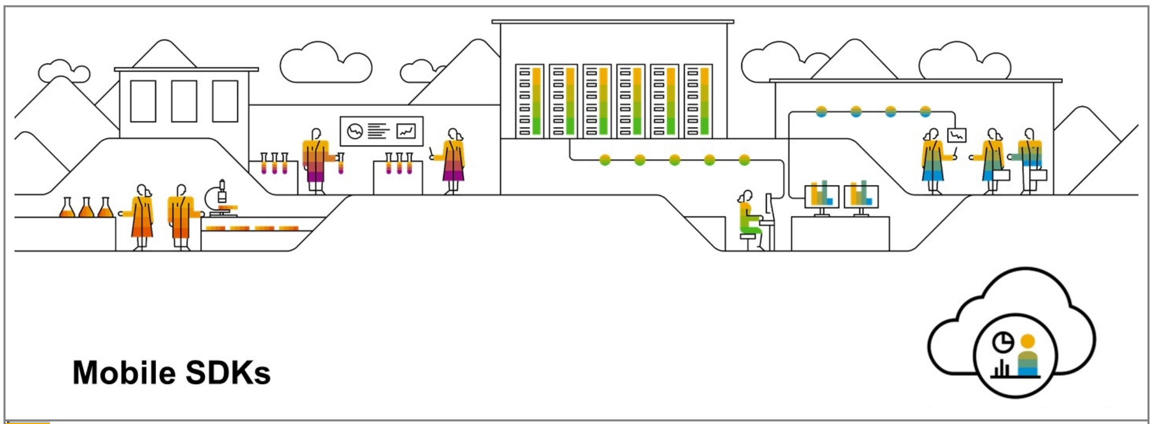
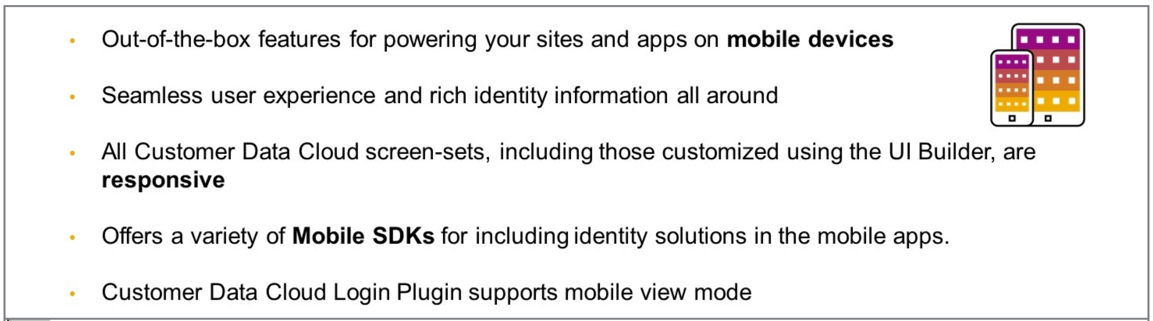


Figure 290: Mobile SDKs



- Out-of-the-box features for powering your sites and apps on **mobile devices**
- Seamless user experience and rich identity information all around
- All Customer Data Cloud screen-sets, including those customized using the UI Builder, are **responsive**
- Offers a variety of **Mobile SDKs** for including identity solutions in the mobile apps.
- Customer Data Cloud Login Plugin supports mobile view mode

Figure 291: Mobile Support

- The Mobile SDKs for **Android** and **iOS** provide an interface for the Customer Data Cloud API from both mobile platforms
- The libraries make it simple to integrate Customer Data Cloud's services in your mobile application
- Provides mobile UIs for SAP Customer Identity
- Data for all Customer Data Cloud plugins can be retrieved via the REST API
- Supporting any 3rd party social networks in the mobile app requires installing the provider's latest native SDK

Figure 292: Mobile SDK

- Providing mobile UIs for RaaS
- Screen-Sets
- Social Login
- Adding connections to Social Networks
- Comments
- Ratings and Reviews
- Share Bars

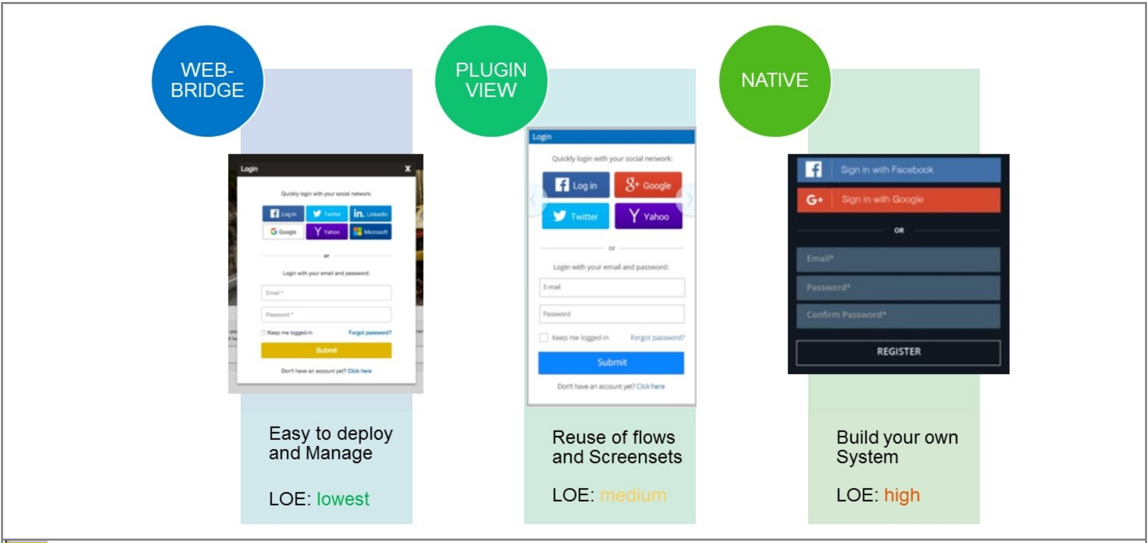
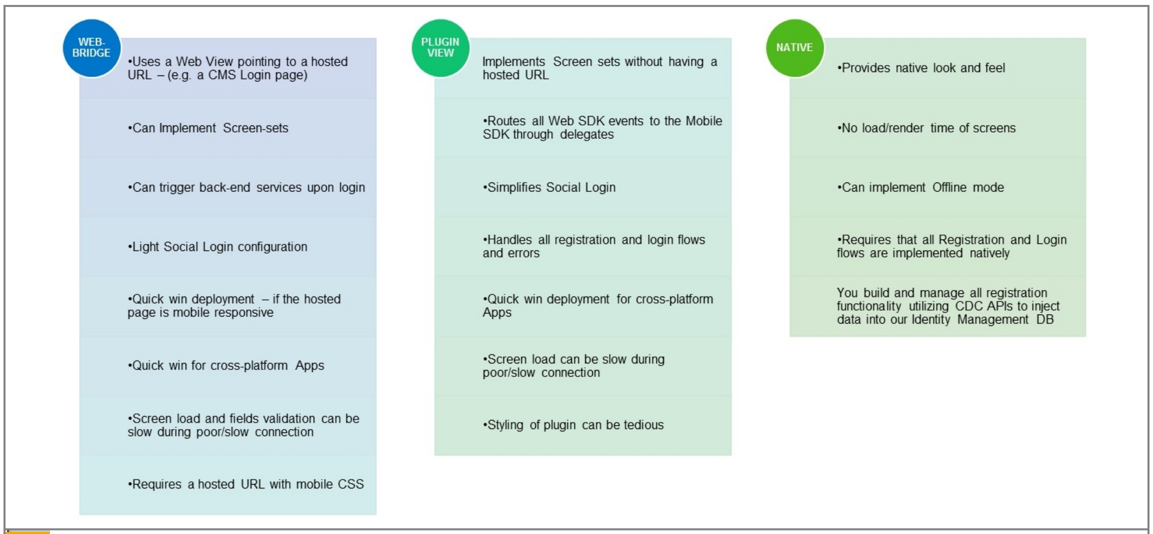


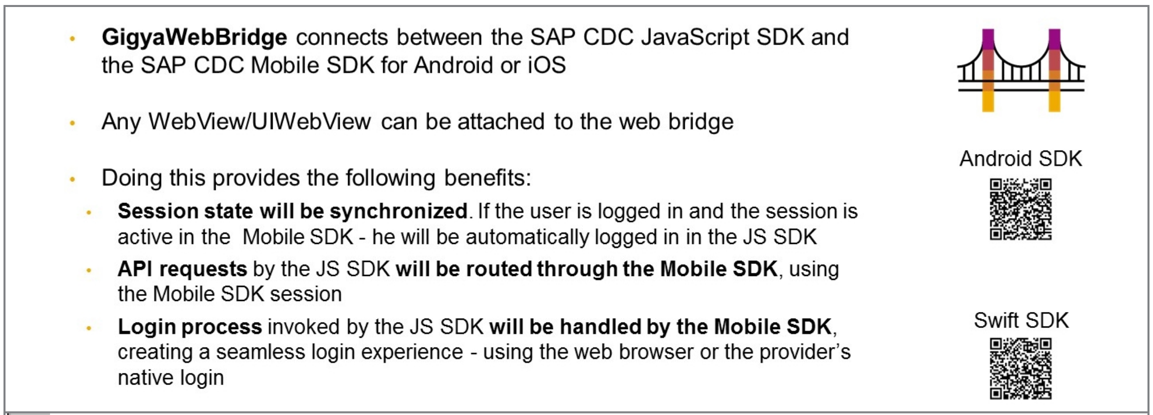
Figure 293: Choosing a Development Approach

- LOE : Level Of Effort



WEB-BRIDGE	PLUGIN VIEW	NATIVE
•Uses a Web View pointing to a hosted URL – (e.g. a CMS Login page)	Implements Screen sets without having a hosted URL	•Provides native look and feel
•Can Implement Screen-sets	•Routes all Web SDK events to the Mobile SDK through delegates	•No load/render time of screens
•Can trigger back-end services upon login	•Simplifies Social Login	•Can implement Offline mode
•Light Social Login configuration	•Handles all registration and login flows and errors	•Requires that all Registration and Login flows are implemented natively
•Quick win deployment – if the hosted page is mobile responsive	•Quick win deployment for cross-platform Apps	You build and manage all registration functionality utilizing CDC APIs to inject data into our Identity Management DB
•Quick win for cross-platform Apps	•Screen load can be slow during poor/slow connection	
•Screen load and fields validation can be slow during poor/slow connection	•Styling of plugin can be tedious	
•Requires a hosted URL with mobile CSS		

Figure 294: Choosing a Development Approach



- **GigyaWebBridge** connects between the SAP CDC JavaScript SDK and the SAP CDC Mobile SDK for Android or iOS
- Any WebView/UIWebView can be attached to the web bridge
- Doing this provides the following benefits:
  - **Session state will be synchronized**. If the user is logged in and the session is active in the Mobile SDK - he will be automatically logged in in the JS SDK
  - **API requests** by the JS SDK **will be routed through the Mobile SDK**, using the Mobile SDK session
  - **Login process** invoked by the JS SDK **will be handled by the Mobile SDK**, creating a seamless login experience - using the web browser or the provider's native login

Android SDK

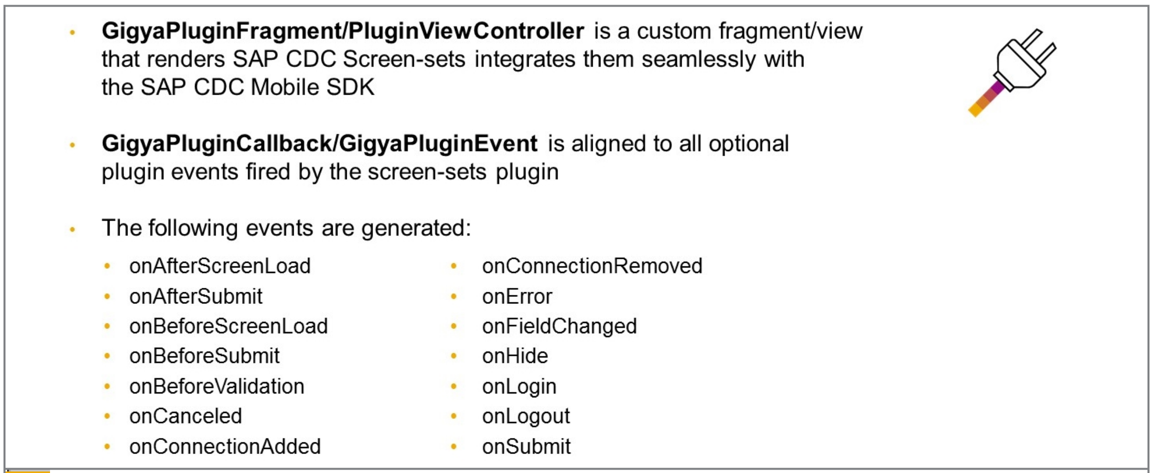
Swift SDK

Figure 295: Web Bridge

Refer to the following links for more details:

https://developers.gigya.com/display/GD/Android+SDK+v4#AndroidSDKv4-UsingtheGigyaWebBridgeexplicitly.

https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-UsingtheGigyaWebBridgeexplicitly.



- **GigyaPluginFragment/PluginViewController** is a custom fragment/view that renders SAP CDC Screen-sets integrates them seamlessly with the SAP CDC Mobile SDK
- **GigyaPluginCallback/GigyaPluginEvent** is aligned to all optional plugin events fired by the screen-sets plugin
- The following events are generated:
  - onAfterScreenLoad
  - onAfterSubmit
  - onBeforeScreenLoad
  - onBeforeSubmit
  - onBeforeValidation
  - onCanceled
  - onConnectionAdded
  - onConnectionRemoved
  - onError
  - onFieldChanged
  - onHide
  - onLogin
  - onLogout
  - onSubmit

Figure 296: Plugin View

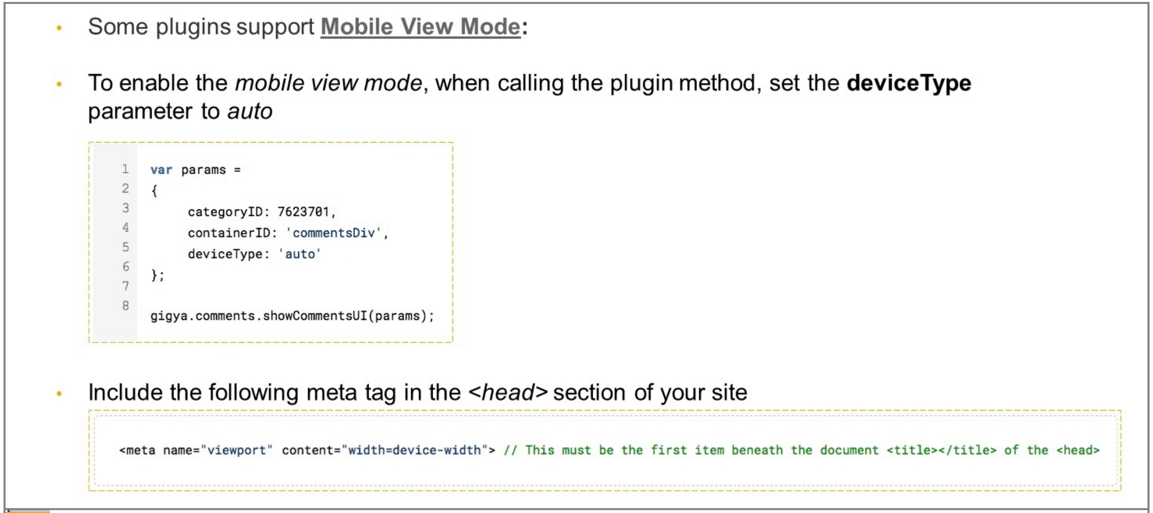
Views vs Fragments

Custom Views have the advantage of simplicity and their primary purpose is to display a piece of data on the screen. They must rely on other components in order to do more.

Think of Fragments as a functional unit, a way to display a portion of UI that has a specific purpose, using one or more Views. Fragments are connected to the Activity lifecycle and they can include and control Loaders to populate the Views with data. They can also include sub-fragments. Finally, they can also be added to a synthetic back stack. They can do many things and are somewhat complex to learn.

As you can see, Fragments have much more in common with Activities than they have with custom views.

As a side note, Fragments can also be headless (with no UI). Headless fragments provide a way to encapsulate non-visual functionality relying on the Activity lifecycle in a separate component.



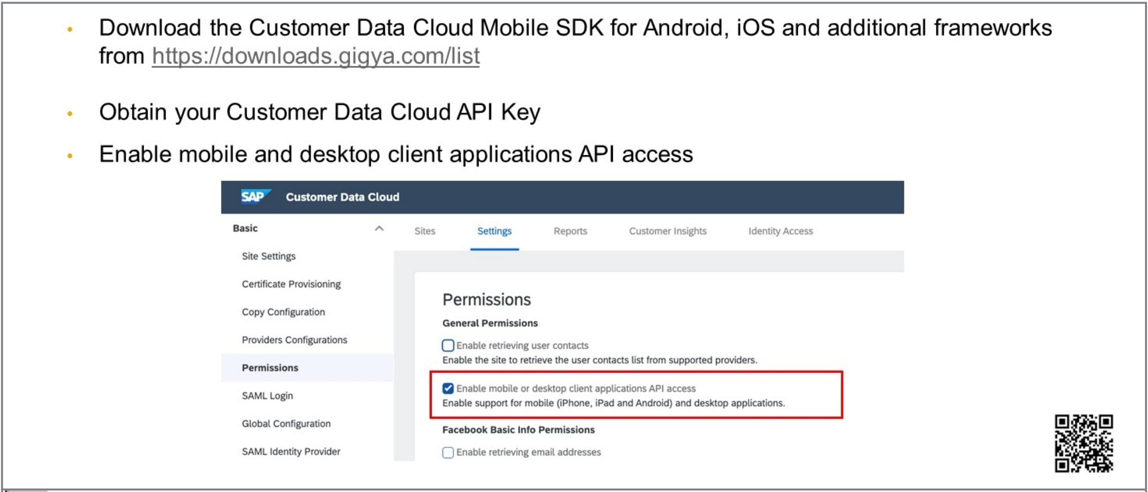
- Some plugins support **Mobile View Mode**:
- To enable the *mobile view mode*, when calling the plugin method, set the **deviceType** parameter to *auto*

```
var params =
{
     categoryID: 7623701,
     containerID: 'commentsDiv',
     deviceType: 'auto'
};

gigya.comments.showCommentsUI(params);
```

- Include the following meta tag in the *<head>* section of your site

```
<meta name="viewport" content="width=device-width"> // This must be the first item beneath the document <title></title> of the <head>
```

Figure 297: Mobile View Mode



- Download the Customer Data Cloud Mobile SDK for Android, iOS and additional frameworks from https://downloads.gigya.com/list
- Obtain your Customer Data Cloud API Key
- Enable mobile and desktop client applications API access

Permissions

General Permissions

- [ ] Enable retrieving user contacts
  Enable the site to retrieve the user contacts list from supported providers.
- [✓] Enable mobile or desktop client applications API access
  Enable support for mobile (iPhone, iPad and Android) and desktop applications.

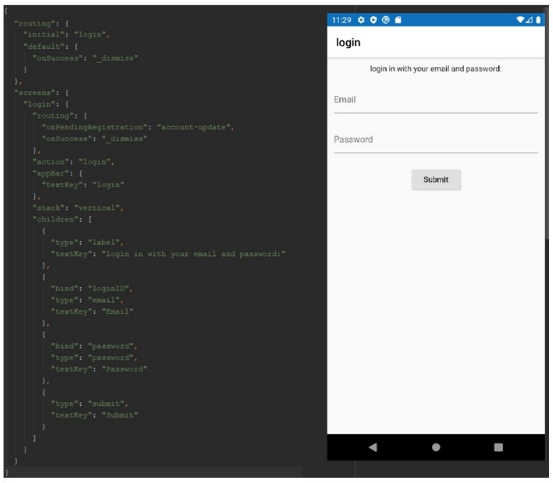
Facebook Basic Info Permissions

- [ ] Enable retrieving email addresses

Figure 298: Download Mobile SDK and Setup Customer Data Cloud

https://downloads.gigya.com/list

- Allows your app to maintain the native experience while enjoying the benefits of SAP Customer Data Cloud web Screen-Sets
- Low-code solution for delivering a highly customizable user interface for a consistent user experience
- Native Screen-Sets are rendered based on a single JSON object
- Login, registration and profile update flows for mobile apps are not only functional but also can enrich the user's experience



```
{
  "routing": {
    "initial": "login",
    "default": {
      "onSuccess": "_dismiss"
    }
  },
  "screens": {
    "login": {
      "routing": {
        "onPendingRegistration": "account-update",
        "onSuccess": "_dismiss"
      },
      "action": "login",
      "appBar": {
        "textKey": "login"
      },
      "stack": "vertical",
      "children": [
        {
          "type": "label",
          "textKey": "login in with your email and password:"
        },
        {
          "bind": "loginID",
          "type": "email",
          "textKey": "Email"
        },
        {
          "bind": "password",
          "type": "password",
          "textKey": "Password"
        },
        {
          "type": "submit",
          "textKey": "Submit"
        }
      ]
    }
  }
}
```

Figure 299: Native Screen-Sets

https://downloads.gigya.com/list

- To support Native Screen-Sets, three SAP Customer Data Cloud libraries need to be implemented in your Android/iOS project.

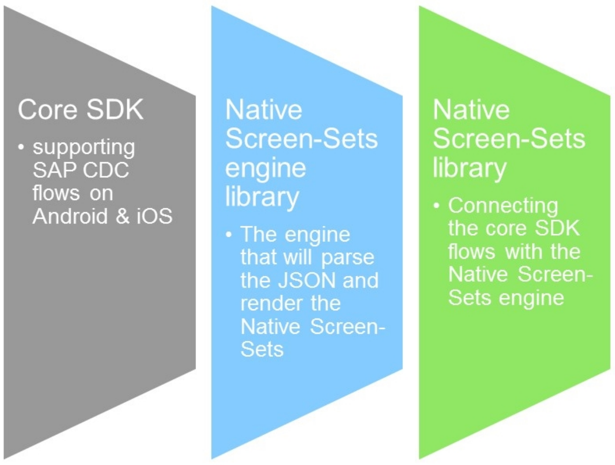
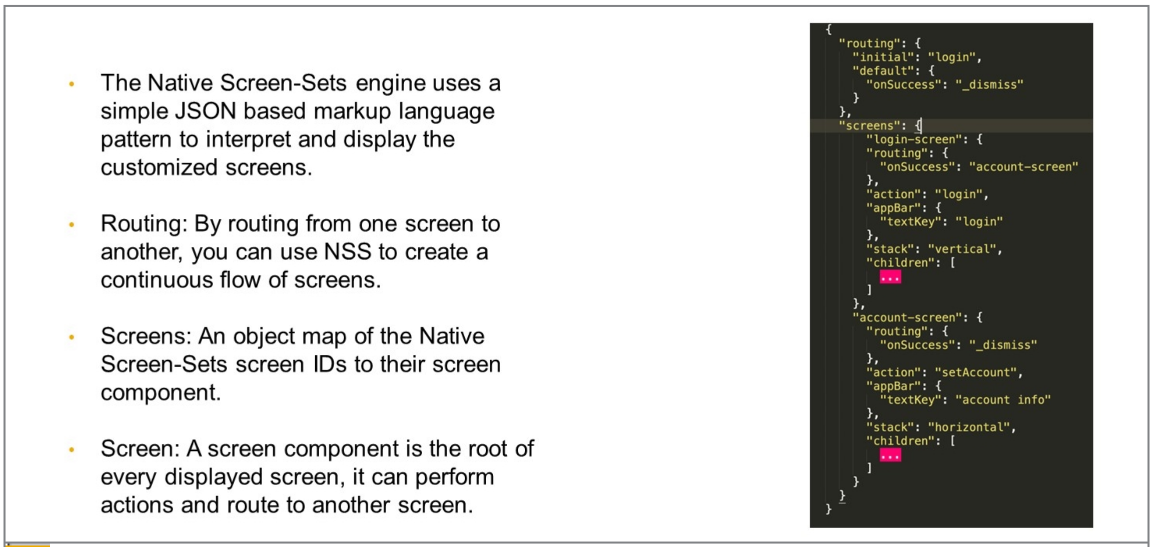

Core SDK
• supporting SAP CDC flows on Android & iOS
Native Screen-Sets engine library
• The engine that will parse the JSON and render the Native Screen-Sets
Native Screen-Sets library
• Connecting the core SDK flows with the Native Screen-Sets engine

Figure 300: Supporting Native Screen-Sets

The engine is developed with Google's Flutter framework



- The Native Screen-Sets engine uses a simple JSON based markup language pattern to interpret and display the customized screens.
- Routing: By routing from one screen to another, you can use NSS to create a continuous flow of screens.
- Screens: An object map of the Native Screen-Sets screen IDs to their screen component.
- Screen: A screen component is the root of every displayed screen, it can perform actions and route to another screen.

```
{
  "routing": {
    "initial": "login",
    "default": {
      "onSuccess": "_dismiss"
    }
  },
  "screens": {
      "login-screen": {
      "routing": {
        "onSuccess": "account-screen"
      },
      "action": "login",
      "appBar": {
        "textKey": "login"
      },
      "stack": "vertical",
      "children": [
        ...
      ]
    },
    "account-screen": {
      "routing": {
        "onSuccess": "_dismiss"
      },
      "action": "setAccount",
      "appBar": {
        "textKey": "account info"
      },
      "stack": "horizontal",
      "children": [
        ...
      ]
    }
  }
}
```

Figure 301: Native Screen-Set JSON Schema

https://downloads.gigya.com/list

Android SDK

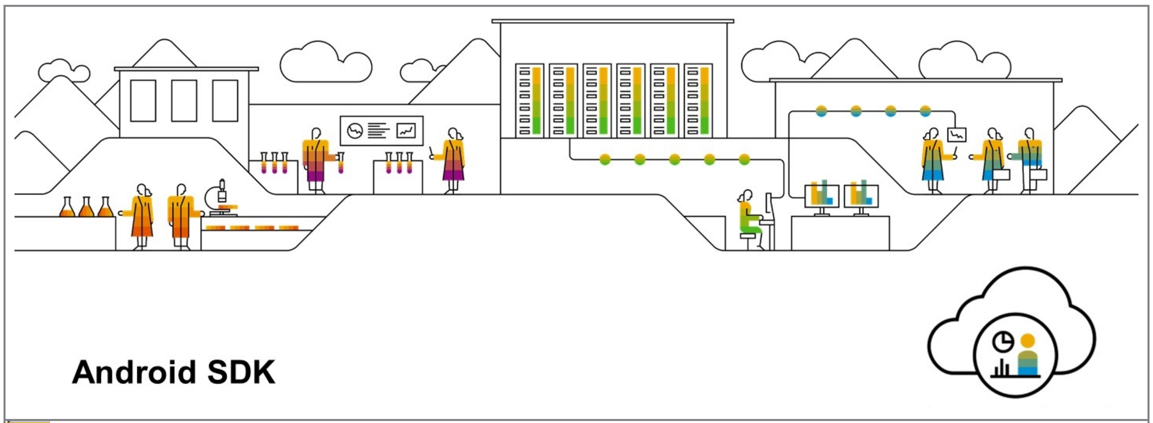
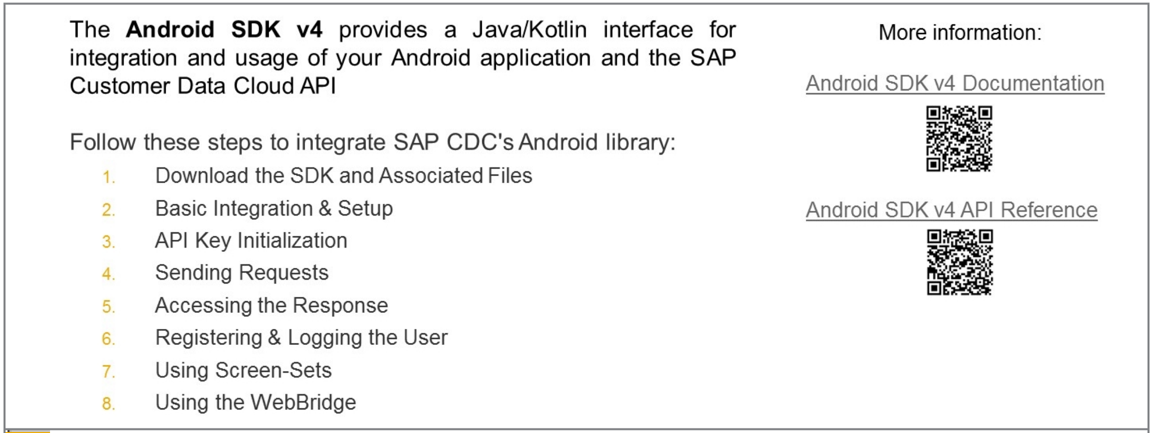


Figure 302: Android SDK



The **Android SDK v4** provides a Java/Kotlin interface for integration and usage of your Android application and the SAP Customer Data Cloud API

Follow these steps to integrate SAP CDC's Android library:

1. Download the SDK and Associated Files
2. Basic Integration & Setup
3. API Key Initialization
4. Sending Requests
5. Accessing the Response
6. Registering & Logging the User
7. Using Screen-Sets
8. Using the WebBridge

More information:

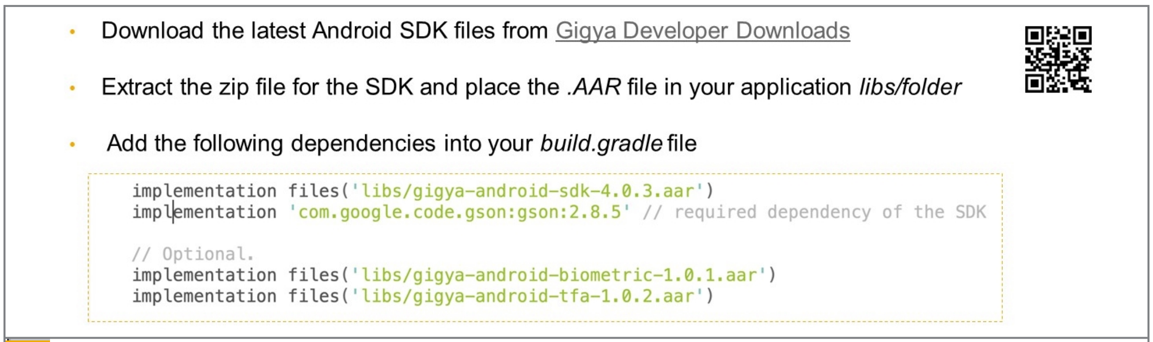
Android SDK v4 Documentation

Android SDK v4 API Reference

Figure 303: Android SDK v4

https://developers.gigya.com/display/GD/Android+SDK+v4

https://developers.gigya.com/display/GD/Android+SDK+v4.x+Reference

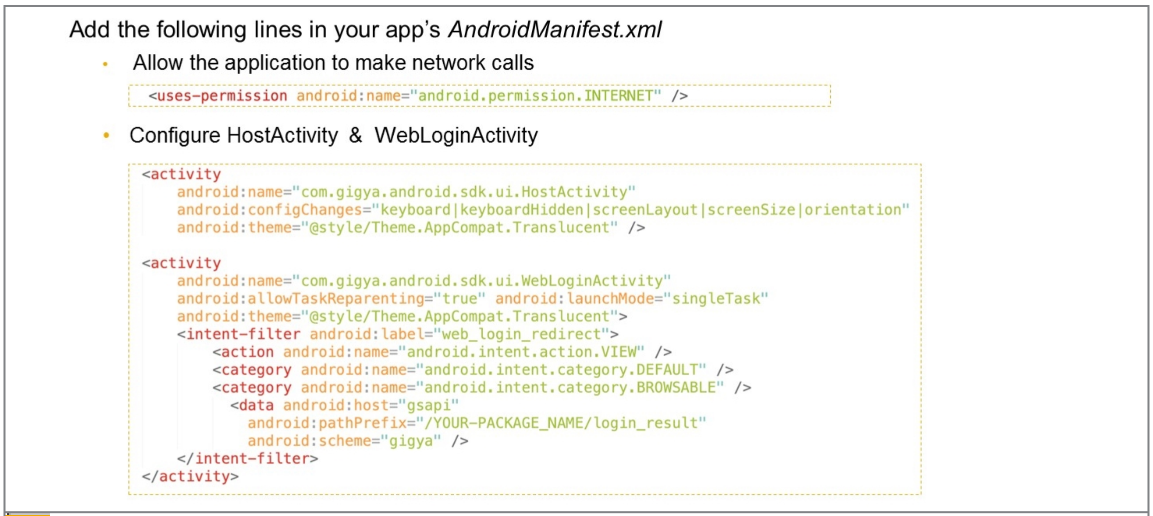


- Download the latest Android SDK files from Gigya Developer Downloads
- Extract the zip file for the SDK and place the *.AAR* file in your application *libs/folder*
- Add the following dependencies into your *build.gradle* file

```
implementation files('libs/gigya-android-sdk-4.0.3.aar')
implementation 'com.google.code.gson:gson:2.8.5' // required dependency of the SDK

// Optional.
implementation files('libs/gigya-android-biometric-1.0.1.aar')
implementation files('libs/gigya-android-tfa-1.0.2.aar')
```

Figure 304: Download the SDK and Associated Files

https://downloads.gigya.com/list



Add the following lines in your app's *AndroidManifest.xml*

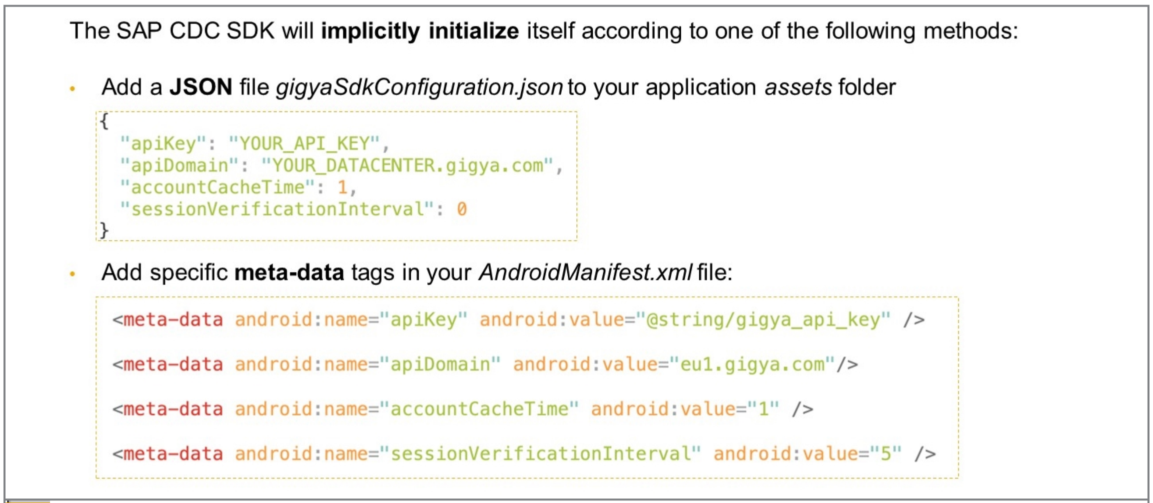
- Allow the application to make network calls

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Configure HostActivity & WebLoginActivity

```
<activity
    android:name="com.gigya.android.sdk.ui.HostActivity"
    android:configChanges="keyboard|keyboardHidden|screenLayout|screenSize|orientation"
    android:theme="@style/Theme.AppCompat.Translucent" />

<activity
    android:name="com.gigya.android.sdk.ui.WebLoginActivity"
    android:allowTaskReparenting="true" android:launchMode="singleTask"
    android:theme="@style/Theme.AppCompat.Translucent">
    <intent-filter android:label="web_login_redirect">
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
          <data android:host="gsapi"
            android:pathPrefix="/YOUR-PACKAGE_NAME/login_result"
            android:scheme="gigya" />
    </intent-filter>
</activity>
```

Figure 305: Basic Integration & Setup



The SAP CDC SDK will **implicitly initialize** itself according to one of the following methods:

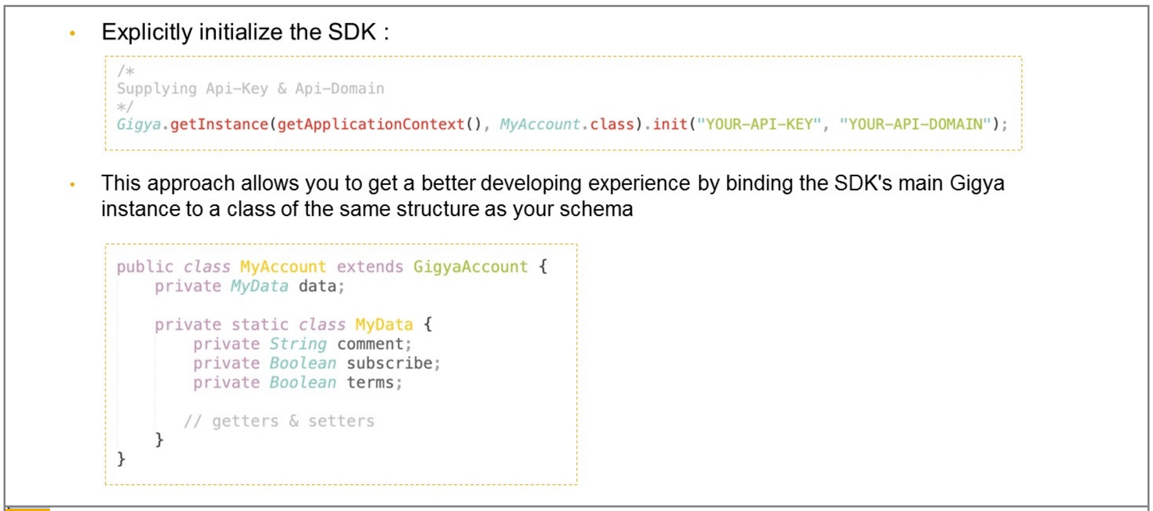
- Add a **JSON** file *gigyaSdkConfiguration.json* to your application *assets* folder

```
{
  "apiKey": "YOUR_API_KEY",
  "apiDomain": "YOUR_DATACENTER.gigya.com",
  "accountCacheTime": 1,
  "sessionVerificationInterval": 0
}
```

- Add specific **meta-data** tags in your *AndroidManifest.xml* file:

```
<meta-data android:name="apiKey" android:value="@string/gigya_api_key" />

<meta-data android:name="apiDomain" android:value="eu1.gigya.com"/>

<meta-data android:name="accountCacheTime" android:value="1" />

<meta-data android:name="sessionVerificationInterval" android:value="5" />
```

Figure 306: Implicit API Key Initialization



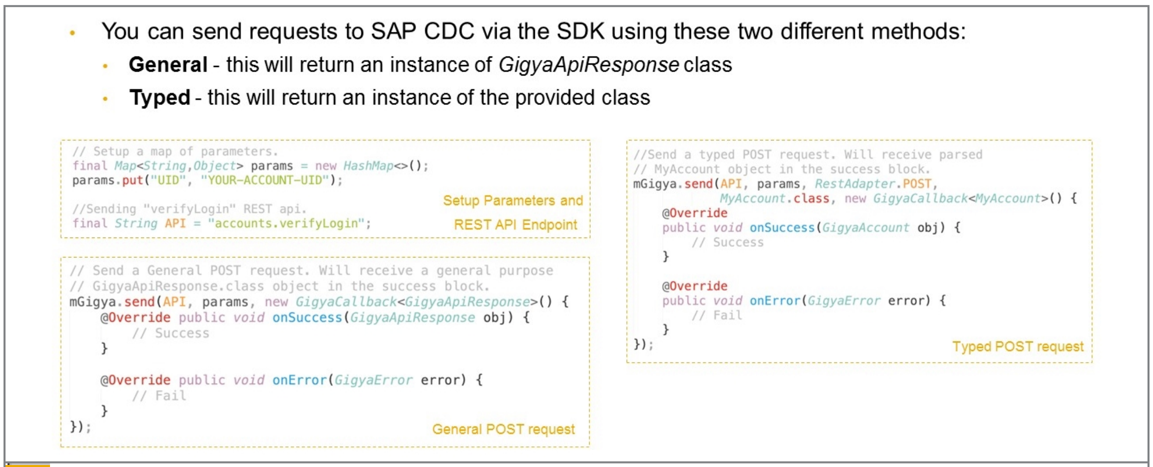
- Explicitly initialize the SDK :

```
/*
Supplying Api-Key & Api-Domain
*/
Gigya.getInstance(getApplicationContext(), MyAccount.class).init("YOUR-API-KEY", "YOUR-API-DOMAIN");
```

- This approach allows you to get a better developing experience by binding the SDK's main Gigya instance to a class of the same structure as your schema

```
public class MyAccount extends GigyaAccount {
    private MyData data;

    private static class MyData {
        private String comment;
        private Boolean subscribe;
        private Boolean terms;

       // getters & setters
    }
}
```

Figure 307: Explicit API Key Initialization



- You can send requests to SAP CDC via the SDK using these two different methods:
  - **General** - this will return an instance of *GigyaApiResponse* class
  - **Typed** - this will return an instance of the provided class

```
// Setup a map of parameters.
final Map<String,Object> params = new HashMap<>();
params.put("UID", "YOUR-ACCOUNT-UID");

//Sending "verifyLogin" REST api.
final String API = "accounts.verifyLogin";
```

Setup Parameters and REST API Endpoint

```
// Send a General POST request. Will receive a general purpose
// GigyaApiResponse.class object in the success block.
mGigya.send(API, params, new GigyaCallback<GigyaApiResponse>() {
    @Override public void onSuccess(GigyaApiResponse obj) {
        // Success
    }

    @Override public void onError(GigyaError error) {
        // Fail
    }
});
```

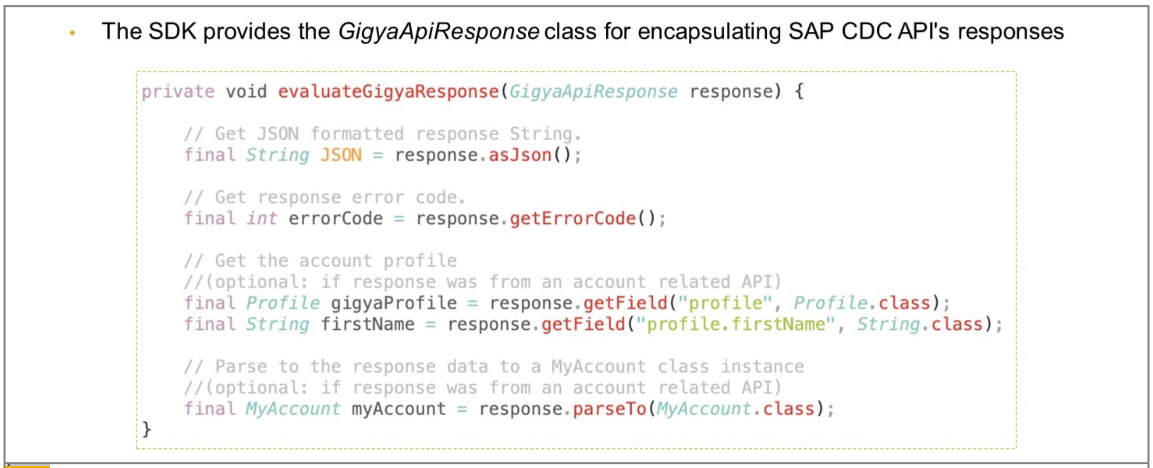
General POST request

```
//Send a typed POST request. Will receive parsed
// MyAccount object in the success block.
mGigya.send(API, params, RestAdapter.POST,
            MyAccount.class, new GigyaCallback<MyAccount>() {
    @Override
    public void onSuccess(GigyaAccount obj) {
        // Success
    }

    @Override
    public void onError(GigyaError error) {
        // Fail
    }
});
```

Typed POST request

Figure 308: Sending Requests

The following example sends an accounts.verifyLogin request using the current logged in user's UID field to verify that the current session is still valid

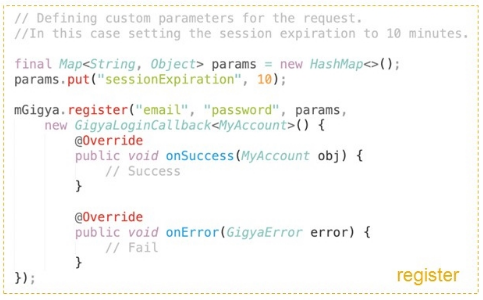


- The SDK provides the *GigyaApiResponse* class for encapsulating SAP CDC API's responses

```
private void evaluateGigyaResponse(GigyaApiResponse response) {

    // Get JSON formatted response String.
    final String JSON = response.asJson();

    // Get response error code.
    final int errorCode = response.getErrorCode();

    // Get the account profile
    //(optional: if response was from an account related API)
    final Profile gigyaProfile = response.getField("profile", Profile.class);
    final String firstName = response.getField("profile.firstName", String.class);

    // Parse to the response data to a MyAccount class instance
    //(optional: if response was from an account related API)
    final MyAccount myAccount = response.parseTo(MyAccount.class);
}
```

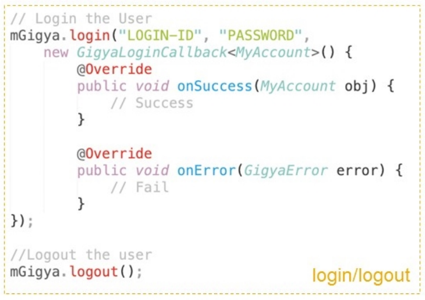
Figure 309: Accessing the Response

- Registration and Site login via API calls is done by using the **register/login** methods. These two methods will end up calling the *GigyaLoginCallback* object



```
// Defining custom parameters for the request.
//In this case setting the session expiration to 10 minutes.

final Map<String, Object> params = new HashMap<>();
params.put("sessionExpiration", 10);

mGigya.register("email", "password", params,
    new GigyaLoginCallback<MyAccount>() {
        @Override
        public void onSuccess(MyAccount obj) {
            // Success
        }

        @Override
        public void onError(GigyaError error) {
            // Fail
        }
});
```

register

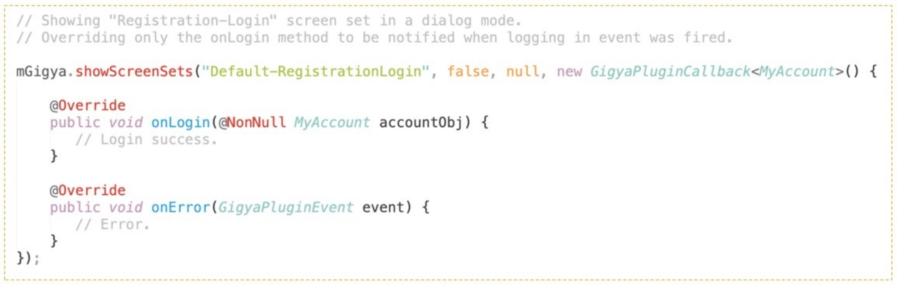


```
// Login the User
mGigya.login("LOGIN-ID", "PASSWORD",
    new GigyaLoginCallback<MyAccount>() {
        @Override
        public void onSuccess(MyAccount obj) {
            // Success
        }

        @Override
        public void onError(GigyaError error) {
            // Fail
        }
});

//Logout the user
mGigya.logout();
```

login/logout

Figure 310: Registering & Logging the User

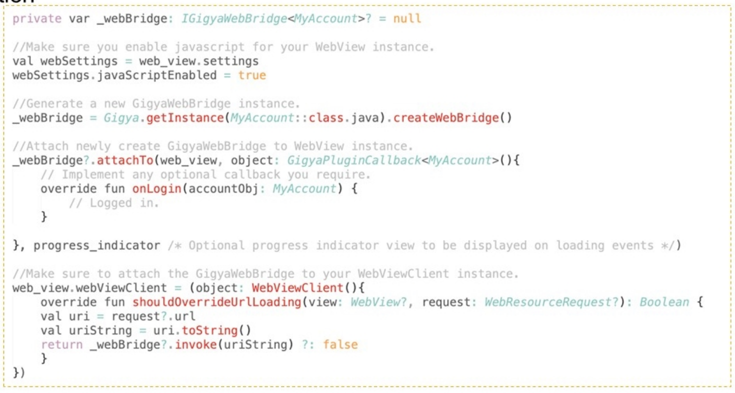
- The SAP CDC Android SDK provides a simple interface for using & displaying screen-sets via the **GigyaPluginFragment** and the **GigyaPluginCallback** components and the *showScreenSets* method of the *Gigya* interface



```
// Showing "Registration-Login" screen set in a dialog mode.
// Overriding only the onLogin method to be notified when logging in event was fired.

mGigya.showScreenSets("Default-RegistrationLogin", false, null, new GigyaPluginCallback<MyAccount>() {

    @Override
    public void onLogin(@NonNull MyAccount accountObj) {
        // Login success.
    }

    @Override
    public void onError(GigyaPluginEvent event) {
        // Error.
    }
});
```

Figure 311: Using Screen-Sets

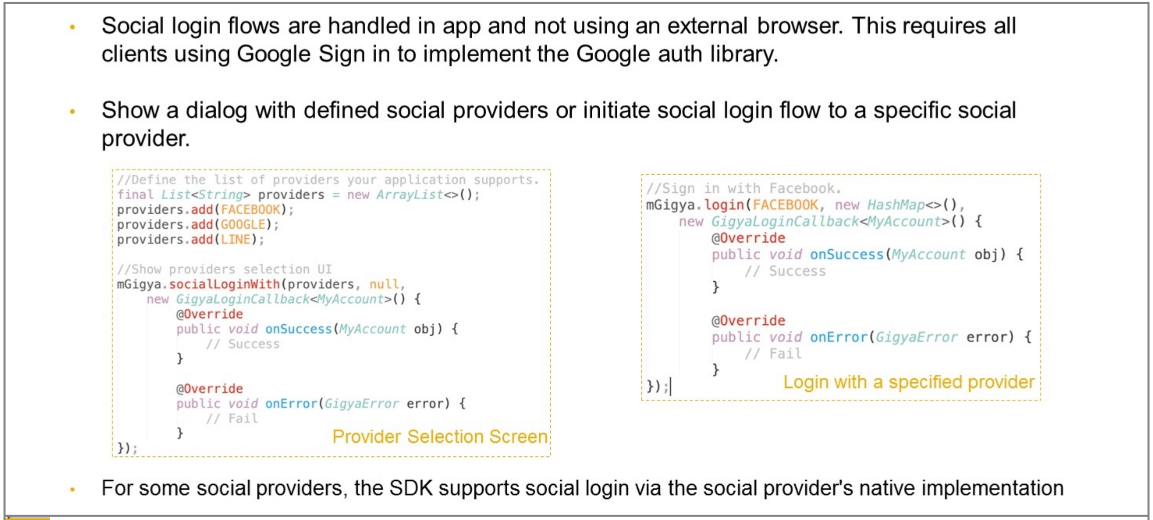
- Use the *GigyaWebBridge* class explicitly to attach CDC's web sdk actions into your own WebView implementation. This will allow you to add session management to your custom web implementation



```
private var _webBridge: IGigyaWebBridge<MyAccount>? = null

//Make sure you enable javascript for your WebView instance.
val webSettings = web_view.settings
webSettings.javaScriptEnabled = true

//Generate a new GigyaWebBridge instance.
_webBridge = Gigya.getInstance(MyAccount::class.java).createWebBridge()

//Attach newly create GigyaWebBridge to WebView instance.
_webBridge?.attachTo(web_view, object: GigyaPluginCallback<MyAccount>(){
    // Implement any optional callback you require.
    override fun onLogin(accountObj: MyAccount) {
        // Logged in.
    }

}, progress_indicator /* Optional progress indicator view to be displayed on loading events */)

//Make sure to attach the GigyaWebBridge to your WebViewClient instance.
web_view.webViewClient = (object: WebViewClient(){
    override fun shouldOverrideUrlLoading(view: WebView?, request: WebResourceRequest?): Boolean {
    val uri = request?.url
    val uriString = uri.toString()
    return _webBridge?.invoke(uriString) ?: false
    }
})
```

Figure 312: Using the WebBridge

Special cases include uses of SAML & captcha implementations



- Social login flows are handled in app and not using an external browser. This requires all clients using Google Sign in to implement the Google auth library.
- Show a dialog with defined social providers or initiate social login flow to a specific social provider.

```
//Define the list of providers your application supports.
final List<String> providers = new ArrayList<>();
providers.add(FACEBOOK);
providers.add(GOOGLE);
providers.add(LINE);

//Show providers selection UI
mGigya.socialLoginWith(providers, null,
    new GigyaLoginCallback<MyAccount>() {
        @Override
        public void onSuccess(MyAccount obj) {
            // Success
        }

        @Override
        public void onError(GigyaError error) {
            // Fail
        }
});
```

Provider Selection Screen

```
//Sign in with Facebook.
mGigya.login(FACEBOOK, new HashMap<>(),
    new GigyaLoginCallback<MyAccount>() {
        @Override
        public void onSuccess(MyAccount obj) {
            // Success
        }

        @Override
        public void onError(GigyaError error) {
            // Fail
        }
});
```

Login with a specified provider

- For some social providers, the SDK supports social login via the social provider's native implementation

Figure 313: Social Login

All supported providers constants are available using GigyaDefinitions.Providers class

More info about social provider's native implementation: https://developers.gigya.com/display/GD/Android+SDK+v4#AndroidSDKv4-ConfiguringNativeLogin

iOS SDK

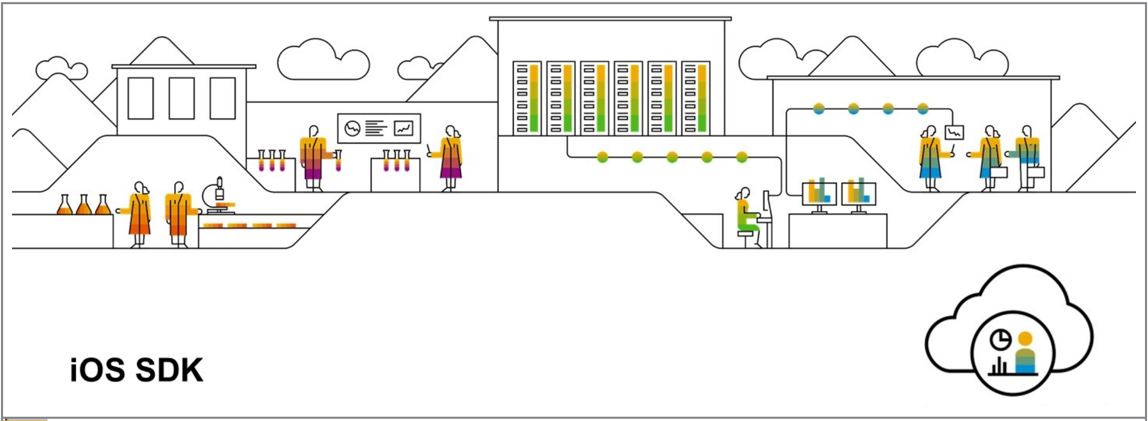
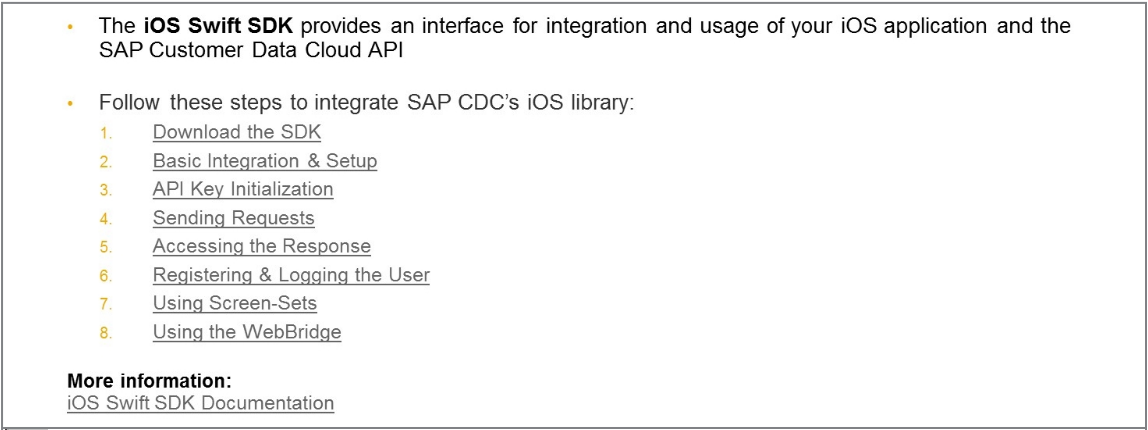


Figure 314: iOS SDK



- The **iOS Swift SDK** provides an interface for integration and usage of your iOS application and the SAP Customer Data Cloud API
- Follow these steps to integrate SAP CDC's iOS library:
  1. Download the SDK
  2. Basic Integration & Setup
  3. API Key Initialization
  4. Sending Requests
  5. Accessing the Response
  6. Registering & Logging the User
  7. Using Screen-Sets
  8. Using the WebBridge

More information:
iOS Swift SDK Documentation

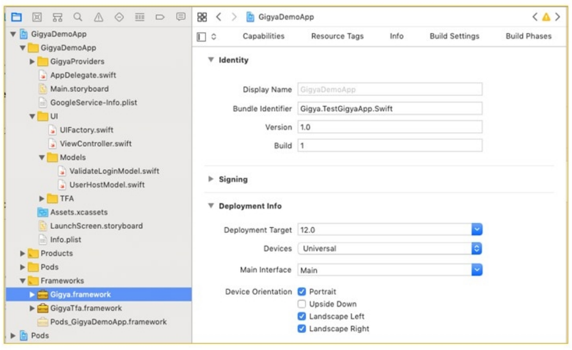
Figure 315: iOS Swift SDK

Follow these steps:

1. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-DownloadSDKandSamples
2. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-BasicIntegration
3. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-ImplicitInitialization
4. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-SendingaRequest
5. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-GigyaApiResultEnumwithAssociatedValues
6. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-Login&Registration
7. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-UsingScreen-Sets
8. https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-UsingtheGigyaWebBridgeexplicitly.

More information: https://developers.gigya.com/display/GD/Swift+SDK

- Download the latest Swift SDK files from Gigya Developer Downloads
- Extract the zip file and add the *Gigya.framework* file into your *Embedded Binaries* and *Linked Frameworks* and *Libraries* sections of your application's Xcode project



GigyaDemoApp — Identity

Display Name: GigyaDemoApp
Bundle Identifier: Gigya.TestGigyaApp.Swift
Version: 1.0
Build: 1

Deployment Info

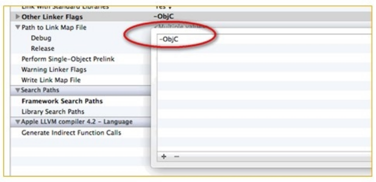
Deployment Target: 12.0
Devices: Universal
Main Interface: Main
Device Orientation: ✓ Portrait, ☐ Upside Down, ✓ Landscape Left, ✓ Landscape Right

```
▼ GigyaDemoApp
  ▼ GigyaDemoApp
    ▶ GigyaProviders
      AppDelegate.swift
      Main.storyboard
      GoogleService-Info.plist
    ▼ UI
        UIFactory.swift
        ViewController.swift
      ▼ Models
          ValidateLoginModel.swift
          UserHostModel.swift
      ▶ TFA
      Assets.xcassets
      LaunchScreen.storyboard
      Info.plist
  ▶ Products
  ▶ Pods
  ▼ Frameworks
    ▶ Gigya.framework
    ▶ GigyaTfa.framework
      Pods_GigyaDemoApp.framework
▶ Pods
```

Figure 316: Download the SDK and Associated Files

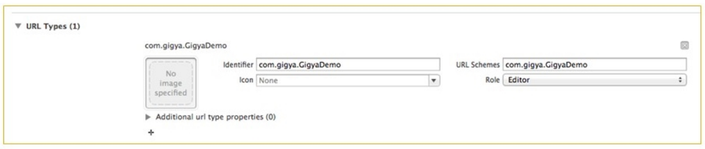
Gigya Developer Downloads: https://downloads.gigya.com/list

- Set the *Build Settings / Other Linker Flags* parameter to: -ObjC



Other Linker Flags: -ObjC

- In app's Info tab, add a *URL type* and enter your *bundle ID* as the identifier and scheme:



URL Types (1)

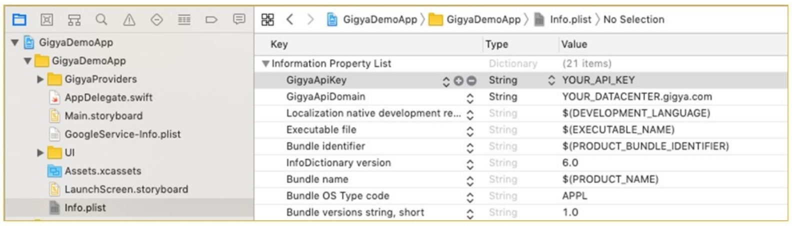
com.gigya.GigyaDemo

Identifier: com.gigya.GigyaDemo
Icon: None
URL Schemes: com.gigya.GigyaDemo
Role: Editor

Figure 317: Basic Integration & Setup (Xcode Project Configuration)

The SDK will implicitly initialize itself according to *Info.plist* configuration.

- Add the following two key/values to your app's *info.plist* file:
  - GigyaApiKey
  - GigyaApiDomain



GigyaDemoApp 〉 GigyaDemoApp 〉 Info.plist 〉 No Selection

- GigyaDemoApp
  - GigyaDemoApp
    - GigyaProviders
    - AppDelegate.swift
    - Main.storyboard
    - GoogleService-Info.plist
    - UI
    - Assets.xcassets
    - LaunchScreen.storyboard
    - Info.plist

Key	Type	Value
Information Property List	Dictionary	(21 items)
GigyaApiKey	String	YOUR_API_KEY
GigyaApiDomain	String	YOUR_DATACENTER.gigya.com
Localization native development re...	String	$(DEVELOPMENT_LANGUAGE)
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0

Figure 318: Implicit API Key Initialization (Xcode Project Configuration)

You can also **explicitly initialize** the SDK :

```
// Using default domain (us1-gigya.com).
Gigya.sharedInstance().initFor(apiKey: "YOUR-API-KEY")

// Supplying Api-Key & Api-Domain
Gigya.sharedInstance().initFor(apiKey: "YOUR-API-KEY", apiDomain: "YOUR-API-DOMAIN")

// Passing the Schema Customizations
Gigya.sharedInstance(UserHost.self).initFor(apiKey: "YOUR-API-KEY")
```

- This approach allows you to get a better developing experience by binding the SDK's main Gigya instance to a class of the same structure as your schema to accept and return account instances according to your specification

```
struct UserHost: GigyaAccountProtocol {
    var UID: String?

    var profile: GigyaProfile?

    var UIDSignature: String?

    // More fields

    var data: MyData?
}

struct MyData: Codable {
    var subscribe: Bool?
    var terms: Bool?
    var comment: String?
}
```

Figure 319: Explicit API Key Initialization

- You can send requests to SAP CDC via the SDK using these two overloads:
  - **General** - this will return a dictionary
  - **Typed** - this will return an instance of the provided class

```
// Setup a map of parameters.
let params = ["UID": "YOUR-ACCOUNT-UID"]

// Sending "verifyLogin" REST api.
let api = "accounts.verifyLogin";
```

Setup Parameters and REST API Endpoint

```
// Send a POST request. Will receive a general purpose
// Dictionary object in the success block.
gigya.send(api: api, params: params) {
    (result) in
        switch result {
        case .success(let data):
            // Success - data is Dictionary
        case .failure(let error):
            break
        }
}
```

General POST request

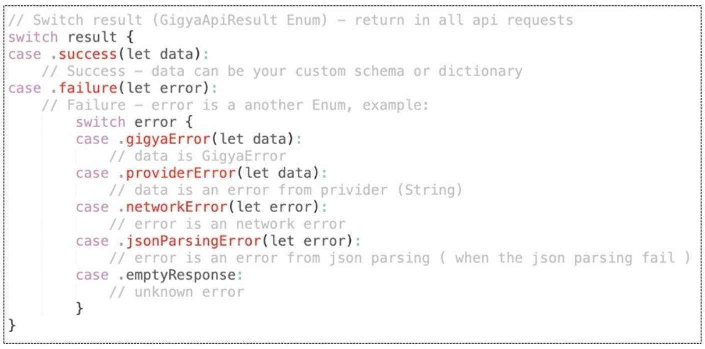
```
// Send a typed POST request. Will receive parsed
// MyAccount object in the success block.
gigya.send(dataType: MyAccount.self, api: api, params: params) {
    (result) in
        switch result {
        case .success(let data):
            // Success - data is MyAccount
        case .failure(let error):
            break
        }
}
```

Typed POST request

Figure 320: Sending Requests

The following example sends an accounts.verifyLogin request using the current logged in user's UID field to verify that the current session is still valid

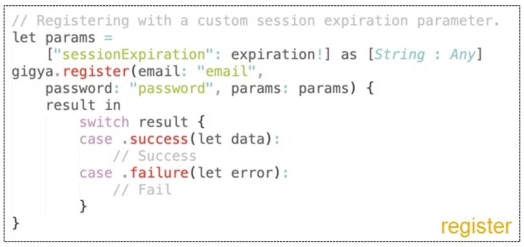
- The SDK provides the *GigyaApiResult* class for encapsulating SAP CDC API's responses



```
// Switch result (GigyaApiResult Enum) - return in all api requests
switch result {
case .success(let data):
    // Success - data can be your custom schema or dictionary
case .failure(let error):
    // Failure - error is a another Enum, example:
        switch error {
        case .gigyaError(let data):
            // data is GigyaError
        case .providerError(let data):
            // data is an error from privider (String)
        case .networkError(let error):
            // error is an network error
        case .jsonParsingError(let error):
            // error is an error from json parsing ( when the json parsing fail )
        case .emptyResponse:
            // unknown error
        }
}
```

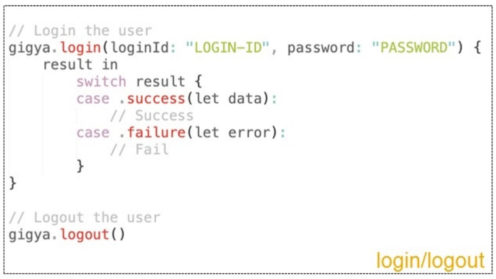
Figure 321: Accessing the Response

- Registration and Site login via API calls is done by using the **register/login** methods. These two methods will end up calling the *GigyaLoginResult* object



```
// Registering with a custom session expiration parameter.
let params =
    ["sessionExpiration": expiration!] as [String : Any]
gigya.register(email: "email",
    password: "password", params: params) {
    result in
        switch result {
        case .success(let data):
            // Success
        case .failure(let error):
            // Fail
        }
}
```

register

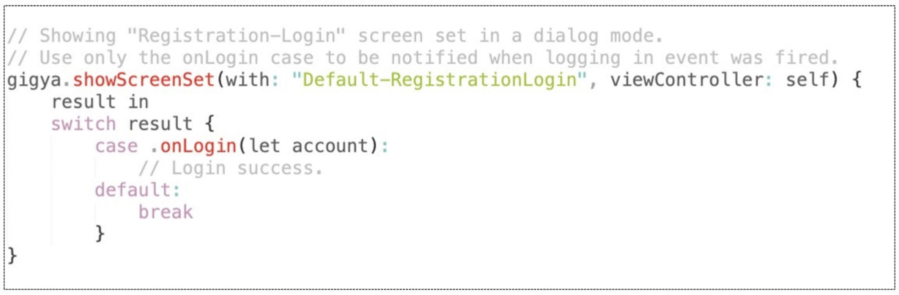


```
// Login the user
gigya.login(loginId: "LOGIN-ID", password: "PASSWORD") {
    result in
        switch result {
        case .success(let data):
            // Success
        case .failure(let error):
            // Fail
        }
}

// Logout the user
gigya.logout()
```

login/logout

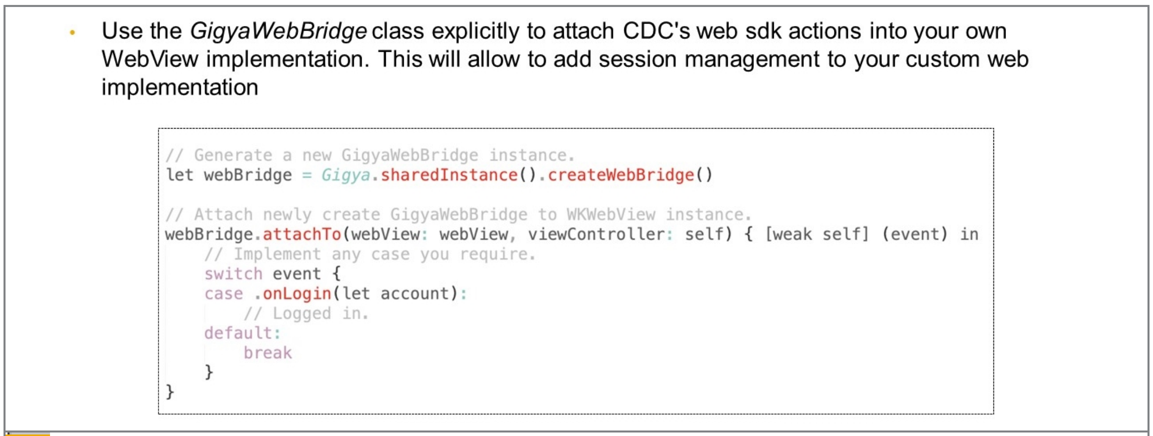
Figure 322: Registering & Logging the User

- The SAP CDC Android SDK provides a simple interface for using & displaying screen-sets via the **PluginViewController** and the **GigyaPluginEvent** components and the *showScreenSet* method of the *Gigya* interface



```
// Showing "Registration-Login" screen set in a dialog mode.
// Use only the onLogin case to be notified when logging in event was fired.
gigya.showScreenSet(with: "Default-RegistrationLogin", viewController: self) {
    result in
    switch result {
        case .onLogin(let account):
            // Login success.
        default:
            break
        }
}
```

Figure 323: Using Screen-Sets

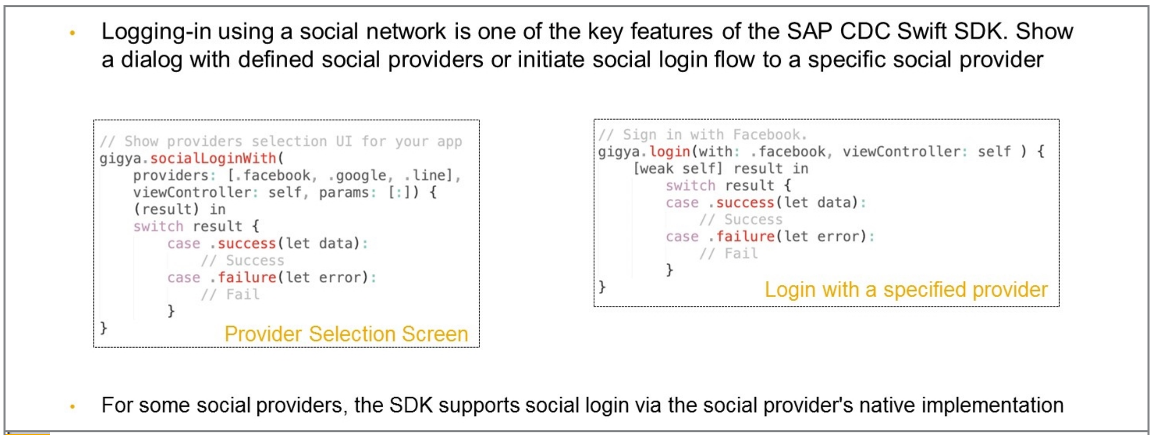


- Use the *GigyaWebBridge* class explicitly to attach CDC's web sdk actions into your own WebView implementation. This will allow to add session management to your custom web implementation

```
// Generate a new GigyaWebBridge instance.
let webBridge = Gigya.sharedInstance().createWebBridge()

// Attach newly create GigyaWebBridge to WKWebView instance.
webBridge.attachTo(webView: webView, viewController: self) { [weak self] (event) in
    // Implement any case you require.
    switch event {
    case .onLogin(let account):
        // Logged in.
    default:
        break
    }
}
```

Figure 324: Using the WebBridge

Special cases include uses of SAML & captcha implementations



- Logging-in using a social network is one of the key features of the SAP CDC Swift SDK. Show a dialog with defined social providers or initiate social login flow to a specific social provider

```
// Show providers selection UI for your app
gigya.socialLoginWith(
    providers: [.facebook, .google, .line],
    viewController: self, params: [:]) {
    (result) in
    switch result {
        case .success(let data):
            // Success
        case .failure(let error):
            // Fail
        }
}
```

Provider Selection Screen

```
// Sign in with Facebook.
gigya.login(with: .facebook, viewController: self ) {
    [weak self] result in
        switch result {
        case .success(let data):
            // Success
        case .failure(let error):
            // Fail
        }
}
```

Login with a specified provider

- For some social providers, the SDK supports social login via the social provider's native implementation

Figure 325: Social Login

All supported providers' constants are available using GigyaSocialProviders Enum.

More info about social providers' native implementation: https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-ConfiguringNativeLogin

Mobile Session Management

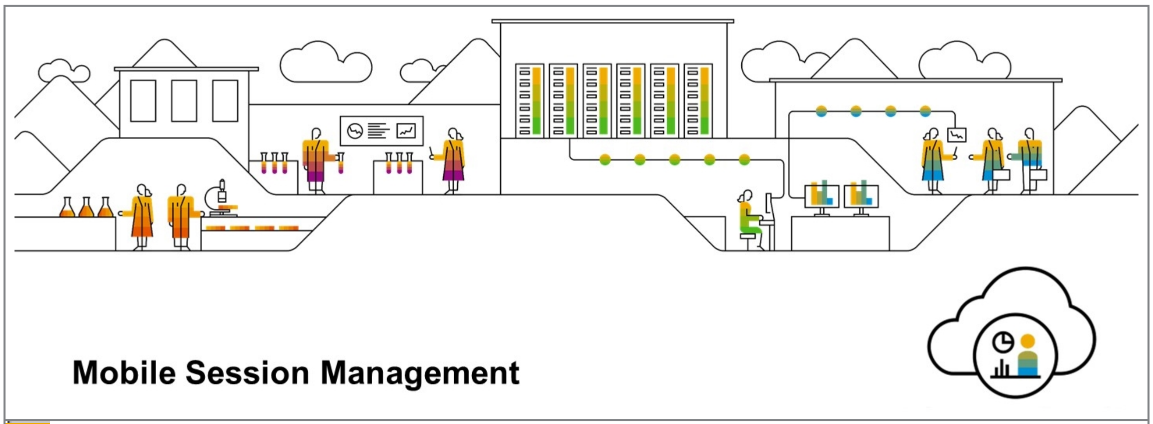
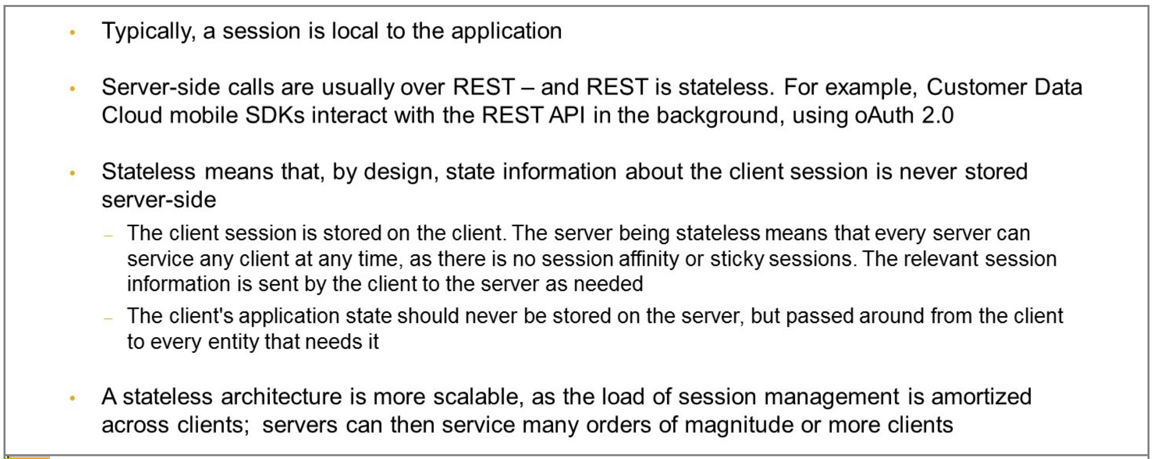
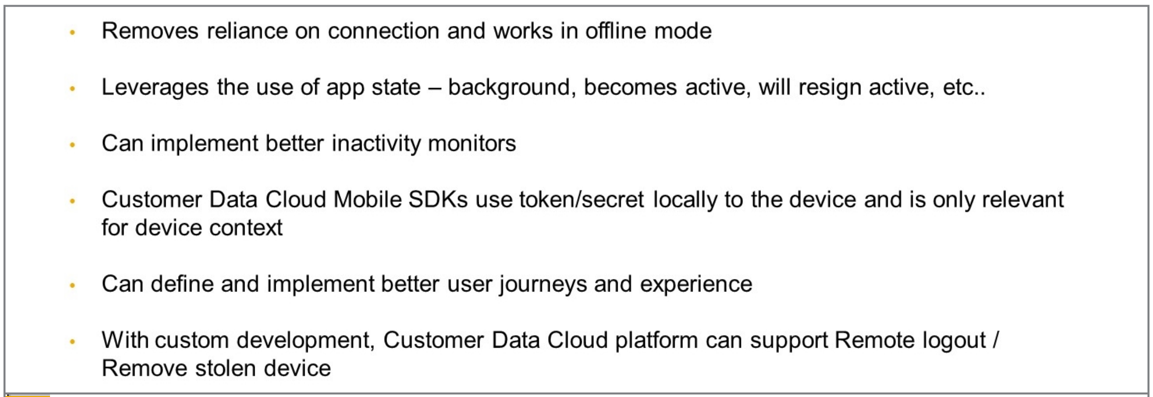


Figure 326: Mobile Session Management



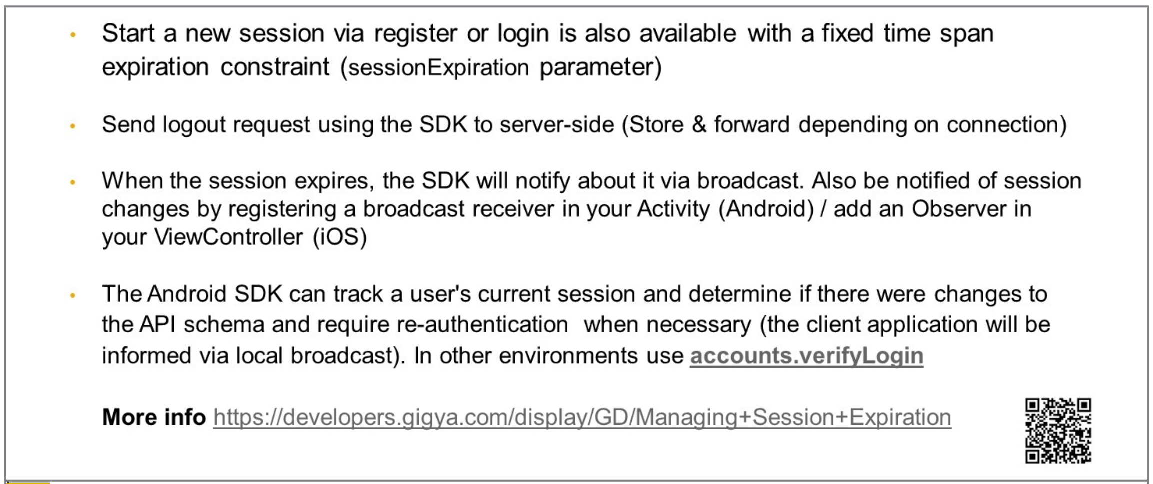
- Typically, a session is local to the application
- Server-side calls are usually over REST – and REST is stateless. For example, Customer Data Cloud mobile SDKs interact with the REST API in the background, using oAuth 2.0
- Stateless means that, by design, state information about the client session is never stored server-side
  - The client session is stored on the client. The server being stateless means that every server can service any client at any time, as there is no session affinity or sticky sessions. The relevant session information is sent by the client to the server as needed
  - The client's application state should never be stored on the server, but passed around from the client to every entity that needs it
- A stateless architecture is more scalable, as the load of session management is amortized across clients; servers can then service many orders of magnitude or more clients

Figure 327: REST and Sessions in Mobile Apps



- Removes reliance on connection and works in offline mode
- Leverages the use of app state – background, becomes active, will resign active, etc..
- Can implement better inactivity monitors
- Customer Data Cloud Mobile SDKs use token/secret locally to the device and is only relevant for device context
- Can define and implement better user journeys and experience
- With custom development, Customer Data Cloud platform can support Remote logout / Remove stolen device

Figure 328: Why Local App sessions



- Start a new session via register or login is also available with a fixed time span expiration constraint (sessionExpiration parameter)
- Send logout request using the SDK to server-side (Store & forward depending on connection)
- When the session expires, the SDK will notify about it via broadcast. Also be notified of session changes by registering a broadcast receiver in your Activity (Android) / add an Observer in your ViewController (iOS)
- The Android SDK can track a user's current session and determine if there were changes to the API schema and require re-authentication when necessary (the client application will be informed via local broadcast). In other environments use **accounts.verifyLogin**

More info https://developers.gigya.com/display/GD/Managing+Session+Expiration

Figure 329: Session Implementation

More info about fixed session duration:

https://developers.gigya.com/display/GD/Android+SDK+v4#AndroidSDKv4-HandlingFixedSessionExpiration

https://developers.gigya.com/display/GD/Swift+SDK#SwiftSDK-FixedLengthSessions

LESSON SUMMARY

You should now be able to:

- Use mobile SDKs and mobile session management

UNIT 12 CIAM for B2B

Lesson 1

UNIT OBJECTIVES

- Use CIAM for B2B

Lesson 12.1: Understanding B2B CIAM

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use CIAM for B2B

CIAM for B2B

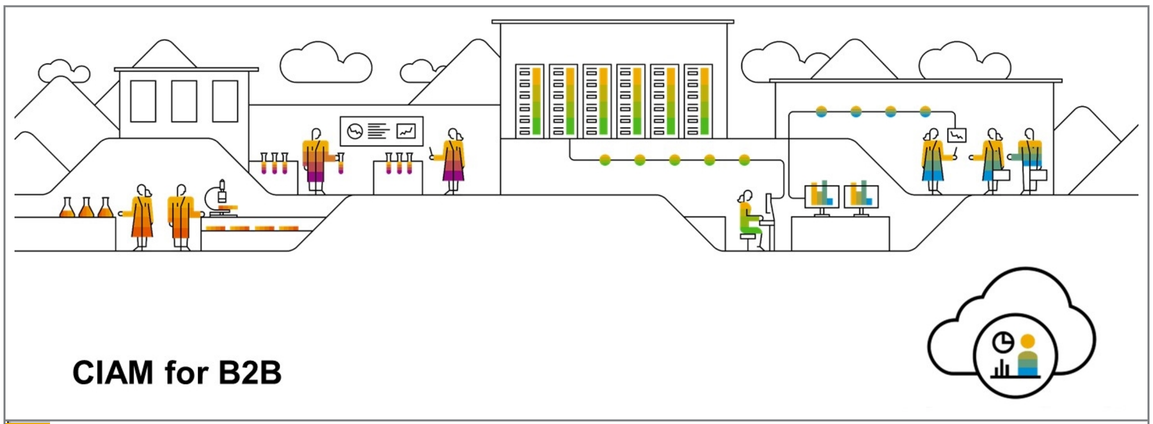


Figure 330: CIAM for B2B

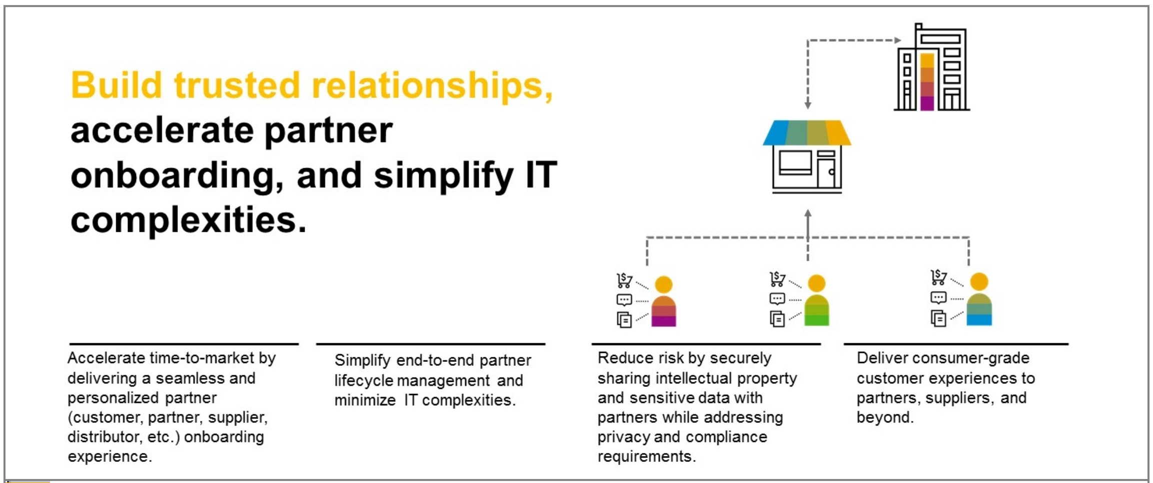
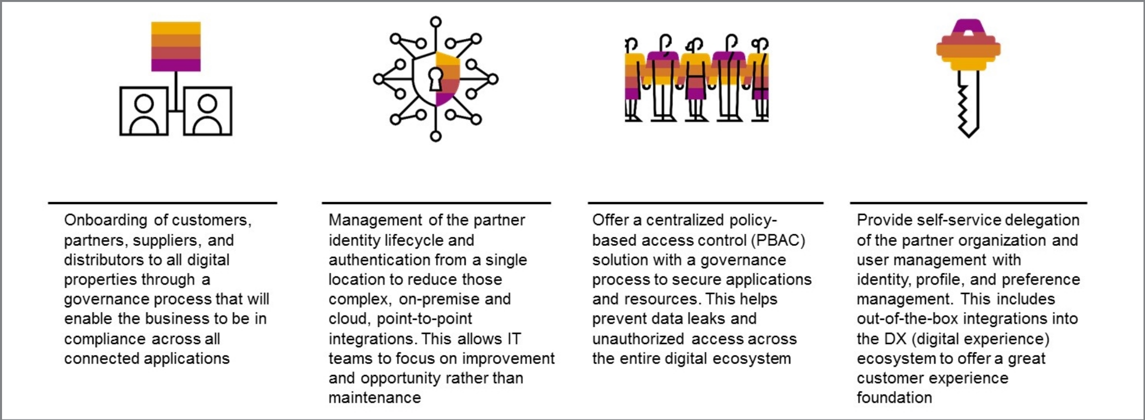

Build trusted relationships, accelerate partner onboarding, and simplify IT complexities.
Accelerate time-to-market by delivering a seamless and personalized partner (customer, partner, supplier, distributor, etc.) onboarding experience.
Simplify end-to-end partner lifecycle management and minimize IT complexities.
Reduce risk by securely sharing intellectual property and sensitive data with partners while addressing privacy and compliance requirements.
Deliver consumer-grade customer experiences to partners, suppliers, and beyond.

Figure 331: Business View and Benefits



Onboarding of customers, partners, suppliers, and distributors to all digital properties through a governance process that will enable the business to be in compliance across all connected applications

Management of the partner identity lifecycle and authentication from a single location to reduce those complex, on-premise and cloud, point-to-point integrations. This allows IT teams to focus on improvement and opportunity rather than maintenance

Offer a centralized policy-based access control (PBAC) solution with a governance process to secure applications and resources. This helps prevent data leaks and unauthorized access across the entire digital ecosystem

Provide self-service delegation of the partner organization and user management with identity, profile, and preference management. This includes out-of-the-box integrations into the DX (digital experience) ecosystem to offer a great customer experience foundation

Figure 332: Key Capabilities

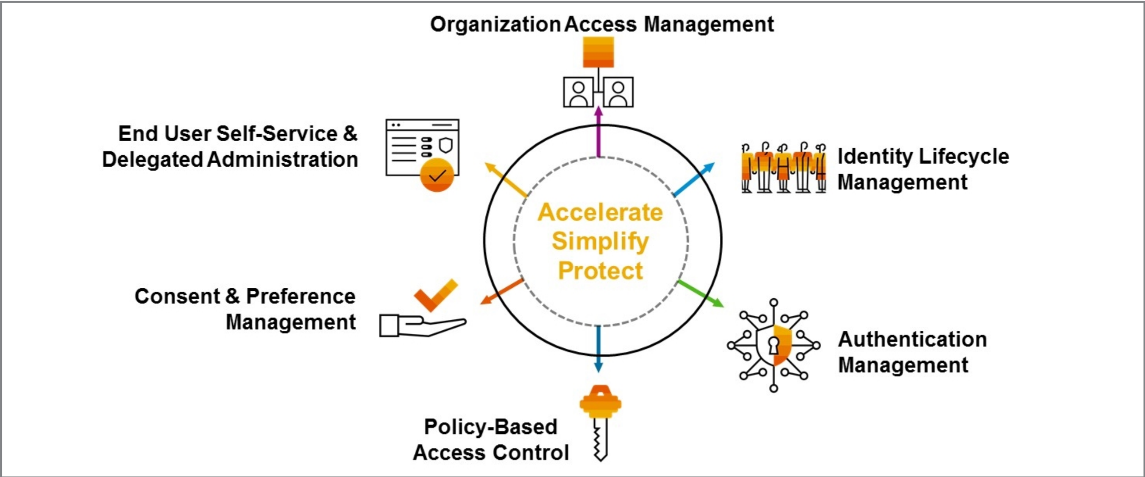

Organization Access Management
End User Self-Service & Delegated Administration
Identity Lifecycle Management
Accelerate
Simplify
Protect
Consent & Preference Management
Authentication Management
Policy-Based Access Control

Figure 333: Introducing SAP CIAM for B2B

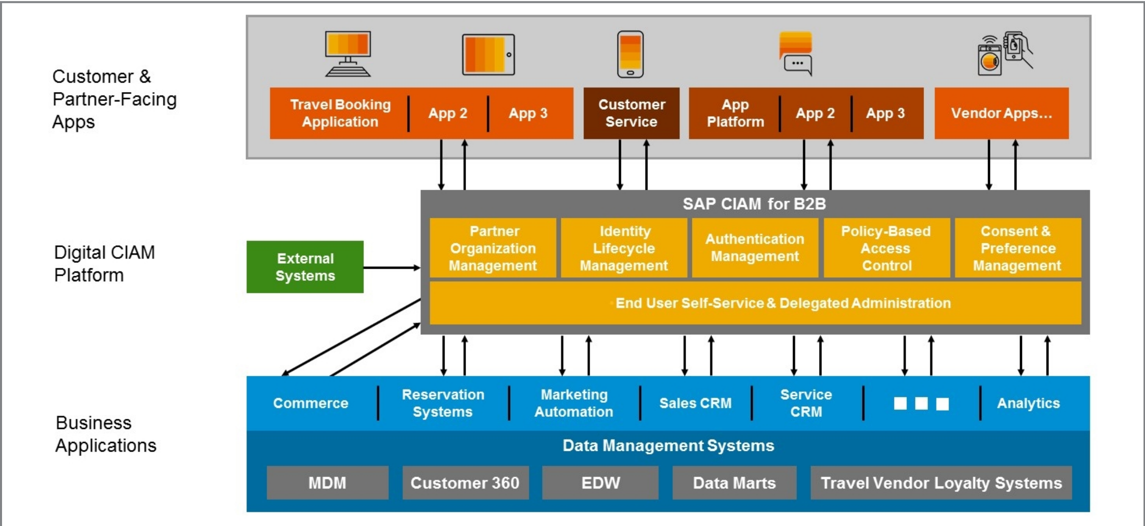
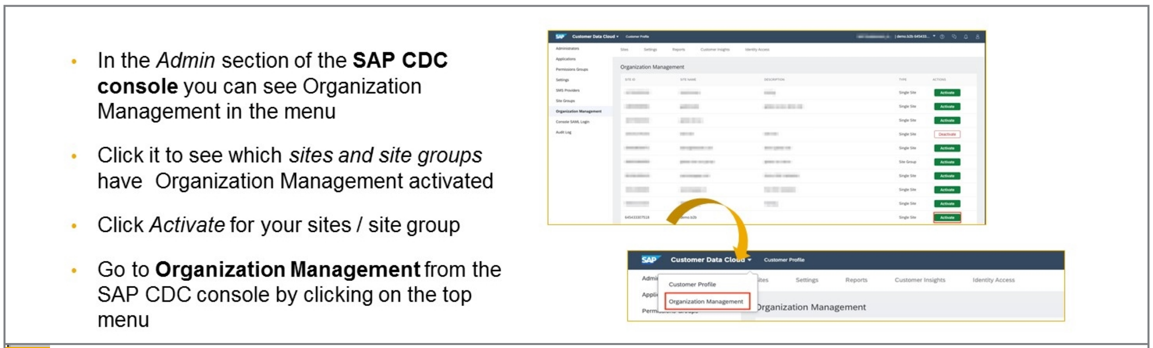

Customer & Partner-Facing Apps
Travel Booking Application
App 2
App 3
Customer Service
App Platform
App 2
App 3
Vendor Apps…
Digital CIAM Platform
External Systems
SAP CIAM for B2B
Partner Organization Management
Identity Lifecycle Management
Authentication Management
Policy-Based Access Control
Consent & Preference Management
End User Self-Service & Delegated Administration
Business Applications
Commerce
Reservation Systems
Marketing Automation
Sales CRM
Service CRM
Analytics
Data Management Systems
MDM
Customer 360
EDW
Data Marts
Travel Vendor Loyalty Systems

Figure 334: Role of SAP CIAM for B2B in the digital architecture

Implementing Partners, Site Groups and Organizations

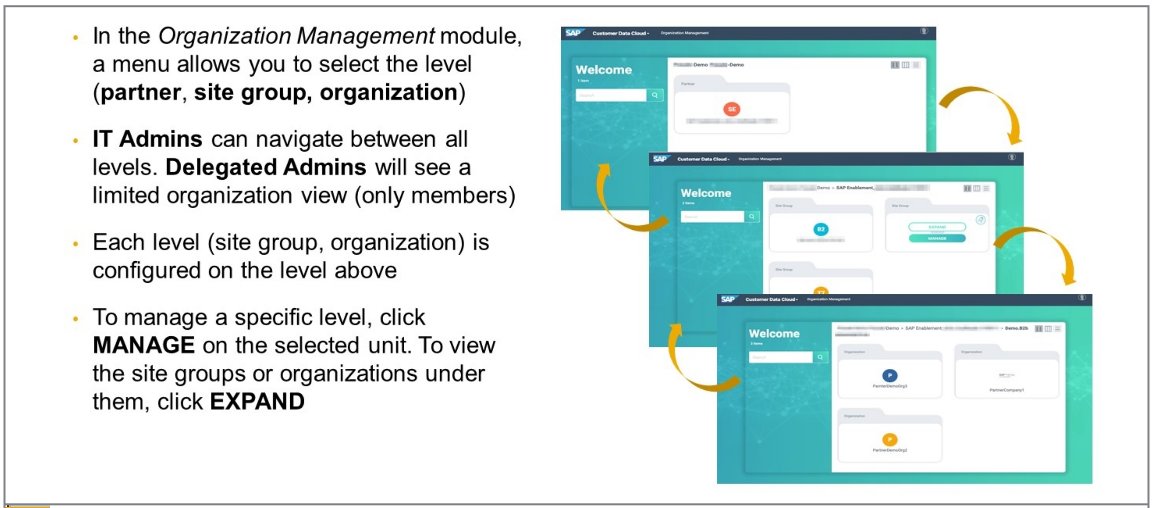
Implementing Partners, Site Groups and Organizations

Figure 335: Implementing Partners, Site Groups and Organizations



- In the *Admin* section of the **SAP CDC console** you can see Organization Management in the menu
- Click it to see which *sites and site groups* have Organization Management activated
- Click *Activate* for your sites / site group
- Go to **Organization Management** from the SAP CDC console by clicking on the top menu

Figure 336: Activate Organization Management and Access to CIAM for B2B Console



- In the *Organization Management* module, a menu allows you to select the level (**partner**, **site group**, **organization**)
- **IT Admins** can navigate between all levels. **Delegated Admins** will see a limited organization view (only members)
- Each level (site group, organization) is configured on the level above
- To manage a specific level, click **MANAGE** on the selected unit. To view the site groups or organizations under them, click **EXPAND**

Figure 337: Organization Management

- **IT Admin** is a CDC user with access to all namespace levels in the Organization Management console
- Capabilities:
  - Manage roles
  - Manage access policies
    - global access policy for partners, site group access policy, and organization access policy
- Manage organization members
  - Invite, update, add/remove roles, status changes, reset password
- Organization management
  - Set up approval workflow, approve/decline organizations

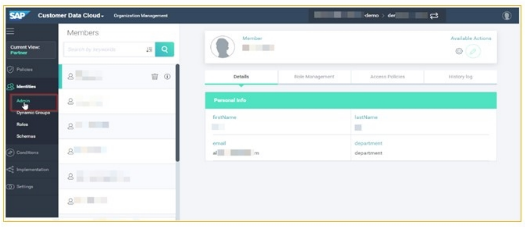
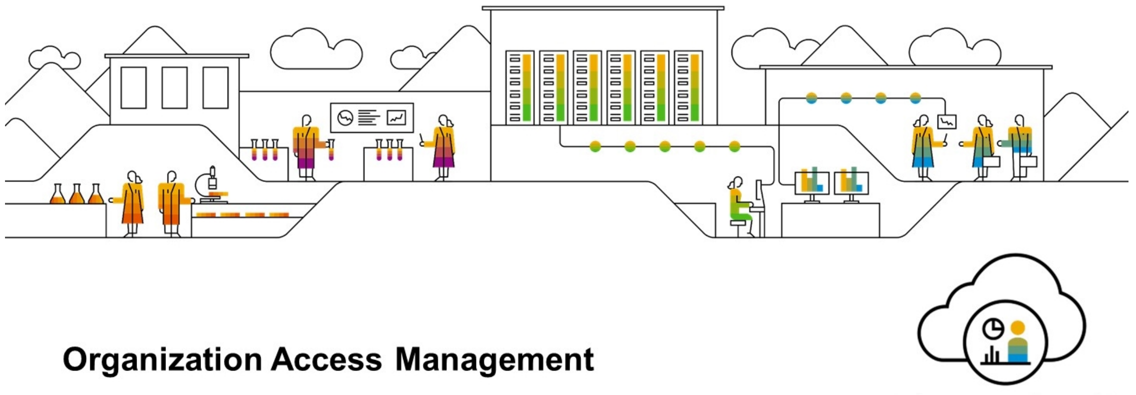


Figure 338: IT Admin Profile

Organization Access Management



Organization Access Management

Figure 339: Organization Access Management

- Facilitates ***rapid onboarding*** of **partner organizations** with capabilities to *create*, *edit*, *update,* and *delete* them
- **Self-registration & provisioning** improves onboarding time and partner experiences with configurable ***approval workflows***
- **User onboarding & management**: *Business Admins* will be able to create and manage *Partner Admins* and *Partner Users*
- **Back office integration**: Automate partner organization synchronization with your back office systems (ERP, CRM, Financial Software, etc.) using an ***open API architecture***

Figure 340: Organization Access Management

- A partner organization can be in different ***statuses***:
  - approved, suspended, in-progress, registration request, etc.
- A business administrator can carry out these ***operations*** on partner organizations:
  - create, remove, edit, suspend, etc.
- A partner organization belongs to an ***organization type***:
  - customer, distributor, vendor, supplier, partner

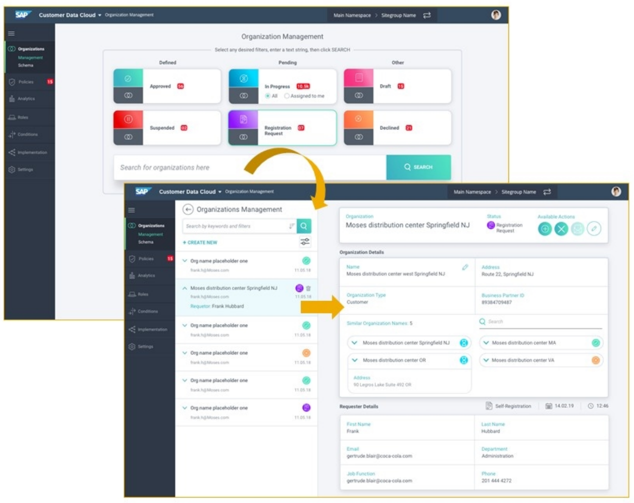


Figure 341: Partner Organizations

- Screen-set for organization self-Registration -> Create a new organization in **Registration Request** status
- A business administrator will assign the organization to an **organization type** and a ***business partner ID (BPID)***
- **Approval workflows** are set up on the *partner* level per *site group* to define the approval path when a new organization is created
- If no workflow is set up, the new organization will be **approved automatically**

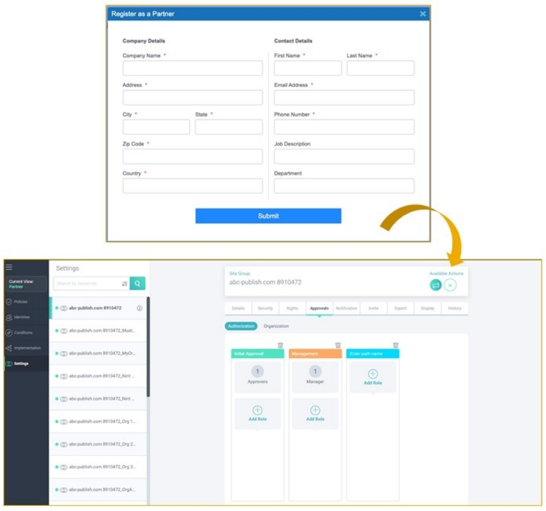


Figure 342: Self-Registration & Provisioning

A **Manage Organizations** RESTful API is available to:

- **Report** about organizations per sites or by BPID
  - **Get roles list by site**
  - **Get list of organizations by site**
  - **Get organization by BPID**
- **Import** multiple organizations

For all APIs, a **bearer token** is required, which needs to be sent as a JWT

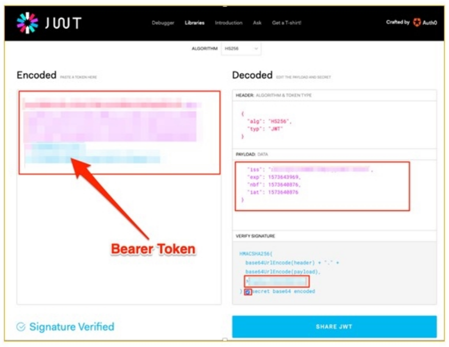


Figure 343: Back Office Integration

BPID stands for Business Partner Id

https://developers.gigya.com/display/GD/Organization+Management+Detailed+User+Guide#OrganizationManagementDetailedUserGuide-4.ManageOrganizationsAPI

End User Self-Service and Delegated Administration

End User Self-Service and Delegated Administration

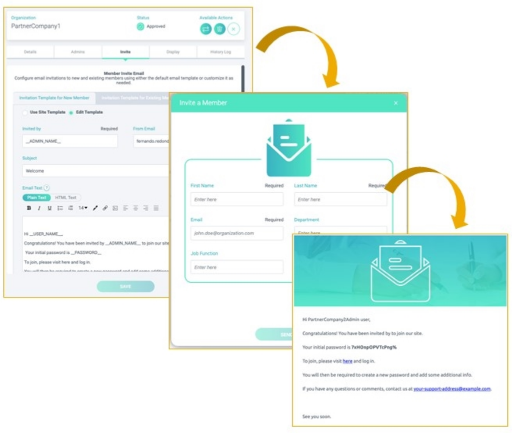
Figure 344: End User Self-Service and Delegated Administration

Provides your partners with the capability to **control their experience** by making changes *without the need to contact you*

- **Partner self-registration**: Makes it easier for your partners to self-register at their convenience rather than waiting for you to *source* them
- **Preference center**: Allows partner users to view their *personal data*, *privacy settings*, and *communication preferences* to then exercise their (GDPR) rights
- **Delegated administration**: Partner administrators can initiate invitation, provisioning, activation, role assignment, password reset, and revocation flows for other users and administrators

Figure 345: End User Self-Service & Delegated Administration

- **Delegated Admin** (*Partner Admin*): A built-in role that can manage members of the organization in a *limited view* of the Organization Management console
- **End Users (***Partner Users): Users* for whom the *authorization requests and responses are* performed (based on predefined roles and policies)
- A **member invitation** email template is defined per site group and can be *customized* for each organization
  - Member invitations include the initial password that each invited user will need to use to log in



Organization: PartnerCompany1 | Status: Approved | Available Actions

Details | Admins | Invite | Display | History Log

Member Invite Email
Configure email invitations to new and existing members using either the default email template or customize it as needed.

Invitation Template for New Member | Invitation Template for Existing Member

Use Site Template ● Edit Template

Invited by (Required): __ADMIN_NAME__
From Email: fernando.redond
Subject: Welcome
Email Text: Plain Text | HTML Text

Hi __USER_NAME__,
Congratulations! You have been invited by __ADMIN_NAME__ to join our site
Your initial password is __PASSWORD__
To join, please visit here and log in.

SAVE

Hi PartnerCompany2Admin user,

Congratulations! You have been invited by to join our site.

Your initial password is **7xHOnpOPVTcPng%**

To join, please visit here and log in.

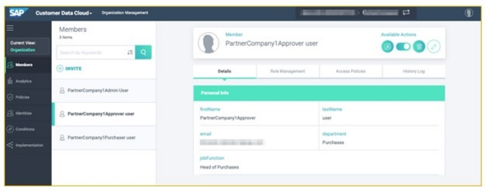
You will then be required to create a new password and add some additional info.

If you have any questions or comments, contact us at your-support-address@example.com.

See you soon.

Figure 346: Partner Users

- Each member who has the **delegated admin** role in the organization can manage organization members (invite, edit, add/remove roles, change status, and reset password)
- All *partner users* from all of the partners are **SAP CDC Identities** for your site group
  - This is accessible for **IT Admins** from the SAP CDC console

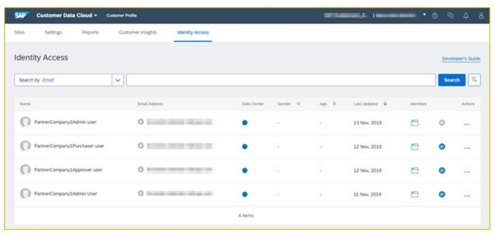


Members
3 Items

- PartnerCompany1Admin User
- PartnerCompany1Approver user
- PartnerCompany1Purchaser user

Member
PartnerCompany1Approver user

firstName: PartnerCompany1Approver	lastName: user
email: [redacted]	department: Purchases
jobFunction: Head of Purchases	



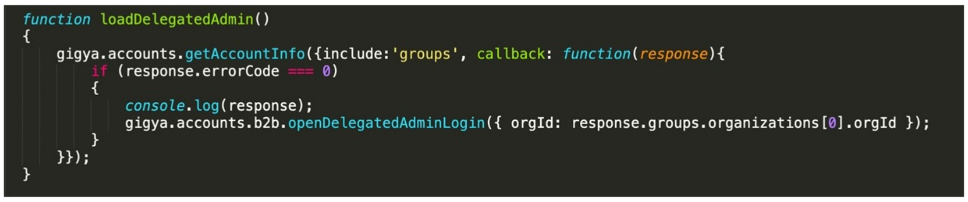
Identity Access

Name	Email Address	Data Center	Gender	Age	Last Updated	Identities	Actions
PartnerCompany2Admin user	[redacted]	●	-	-	13 Nov, 2019		...
PartnerCompany1Purchaser user	[redacted]	●	-	-	12 Nov, 2019		...
PartnerCompany1Approver user	[redacted]	●	-	-	12 Nov, 2019		...
PartnerCompany1Admin User	[redacted]	●	-	-	11 Nov, 2019		...

4 items

Figure 347: Organization Members and Identity Access Views

- This is the implementation in JavaScript for performing the delegated admin's login into the CDC console:



```
function loadDelegatedAdmin()
{
    gigya.accounts.getAccountInfo({include:'groups', callback: function(response){
        if (response.errorCode === 0)
        {
            console.log(response);
            gigya.accounts.b2b.openDelegatedAdminLogin({ orgId: response.groups.organizations[0].orgId });
        }
    }});
}
```

- A new browser tab will appear where the delegated admin can list the current members for the organization and send invitations for more users of the company
- The *organization logo* and the delegated admin's *headline text* can be assigned/updated by the partner manager

Figure 348: Implementing Access for the Delegated Admin

Policy Based Access Control

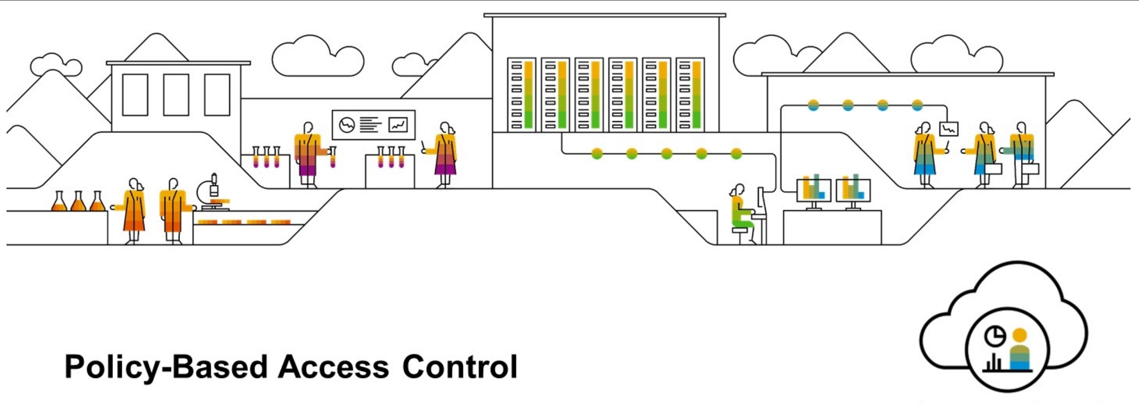


Figure 349: Policy Based Access Control

- An **authorization framework** for digital access with *full policy lifecycle management* for resources that help you to achieve access control compliance
- **Coarse and fine-grained**: Access is determined by assigned roles (RBAC) created with specific permissions or according to the attributes (ABAC) of the requestor, resource, and environment
- **Run-time authorization**: Make access decisions only when needed and when the user is actually *accessing* an application and/or data. This reduces the need for unmanaged AuthZ and ghost IDs

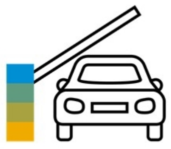


Figure 350: Policy-Based Access Control

RBAC - Role Based Access Control

ABAC - Attribute Based Access Control

- Creating an **access policy** is divided into two main sections:
  I. **Preparing to build a policy** (creating the *policy building blocks*)
     – Once the building blocks are created, use them to build as many policies as needed
  II. **Building an access policy**
- The steps to create the building blocks of a policy are:

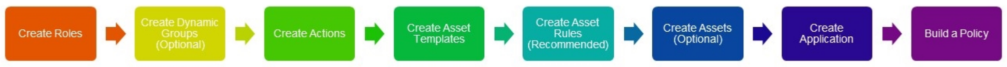
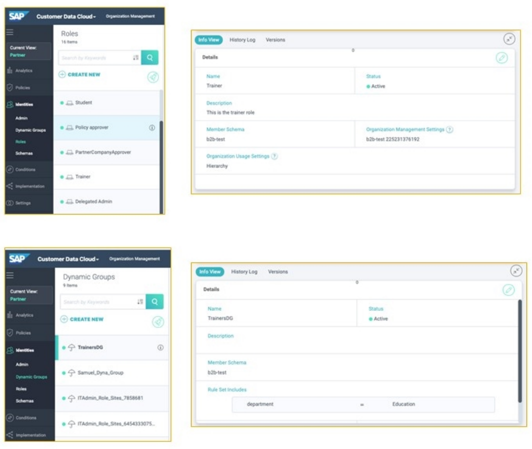


Figure 351: Policy Building Blocks and Access Policy End-to-End

1. Create/manage **roles**
   – A role represents a ***function*** that can be associated with a user (organization member or IT administrator)
   – For example: banker, clerk, doctor, nurse, etc.
2. Create/manage **dynamic groups**
   – A dynamic group represents a collection of users that is built based on their attributes
   – User attributes are sourced from CDC repositories



Roles
16 Items
Student
Policy approver
PartnerCompanyApprover
Trainer
Delegated Admin

Name: Trainer
Status: Active
Description: This is the trainer role
Member Schema: b2b-test
Organization Management Settings: b2b-test 225231376192
Organization Usage Settings: Hierarchy

Dynamic Groups
9 Items
TrainersDG
Samuel_Dyna_Group
ITAdmin_Role_Sites_7858681
ITAdmin_Role_Sites_6454333075...

Name: TrainersDG
Status: Active
Description:
Member Schema: b2b-test
Rule Set Includes: department = Education

Figure 352: Policy Building Blocks I: Roles and Dynamic Groups

3. Create/manage **actions**
   - An action is part of the authorization response. It communicates to the application what is permitted on the requested asset
   - For example: create, read, update, and delete are common actions
4. Create/manage **asset templates**
   - An asset template includes the full list of asset attributes and a full list of possible actions
   - When you build an asset template, you must specify all of the possible actions and attributes to use in your various assets

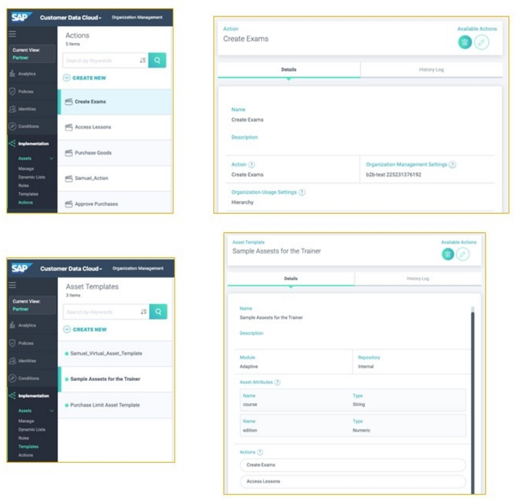


Figure 353: Policy Building Blocks II: Actions and Asset Templates

5. Create/manage **asset rules**
   - An asset rule is an optional filter on the asset that is used in access policies
   - For example: VIP accounts and basic accounts
6. Create/manage **assets**
   - An asset defines upon what the application can act. Each asset is based on a specific asset template. It uses the available asset attributes and possible actions defined in the asset template to create a unique asset in your policy

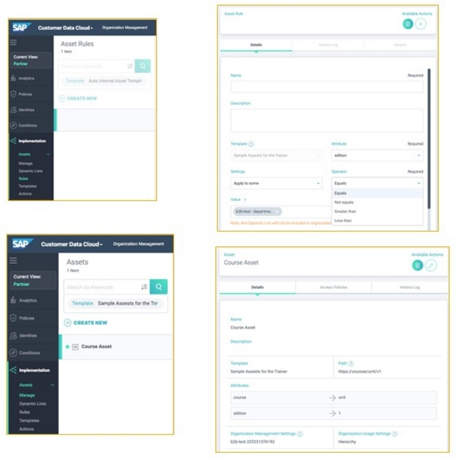


Figure 354: Policy Building Blocks III: Asset Rules and Assets

7. Create/manage **applications**
   - An application represents the component that consumes the authorization information delivered by SAP CDC
   - The application is associated with a unique identifier that is required to submit valid authorization requests. SAP CDC uses this unique identifier to provide the correct authorization for the application

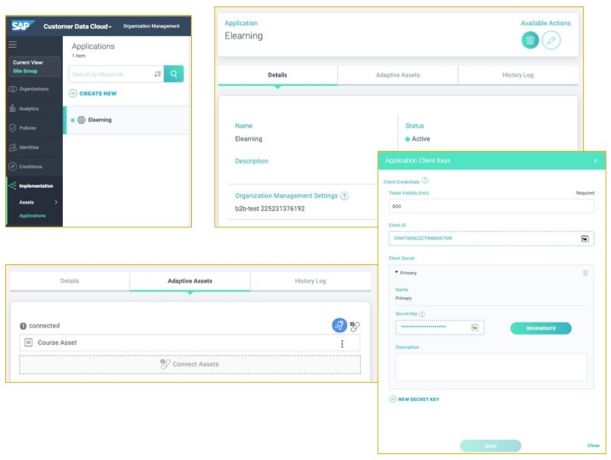
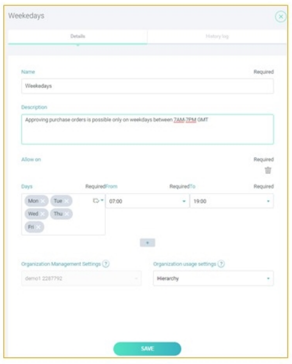


Figure 355: Policy Building Blocks IV: Applications

8. Create/manage **conditions**
   - Conditions place limits on the identities (dynamic groups and roles) in the policy map
   - There are 2 types of conditions:
     - **Time restrictions** (defined in GMT)
     - **Advanced** (IP address, authentication method, additional identity attribute, etc.)


Weekedays

Details | History log

Name (Required): Weekedays

Description: Approving purchase orders is possible only on weekdays between 7AM-7PM GMT

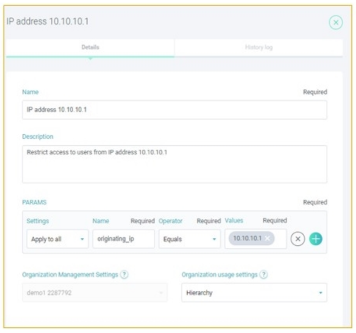
Allow on (Required)

Days (Required): Mon, Tue, Wed, Thu, Fri — From (Required): 07:00 — To (Required): 19:00

Organization Management Settings: demo1 2287792

Organization usage settings: Hierarchy

SAVE


IP address 10.10.10.1

Details | History log

Name (Required): IP address 10.10.10.1

Description: Restrict access to users from IP address 10.10.10.1

PARAMS (Required)

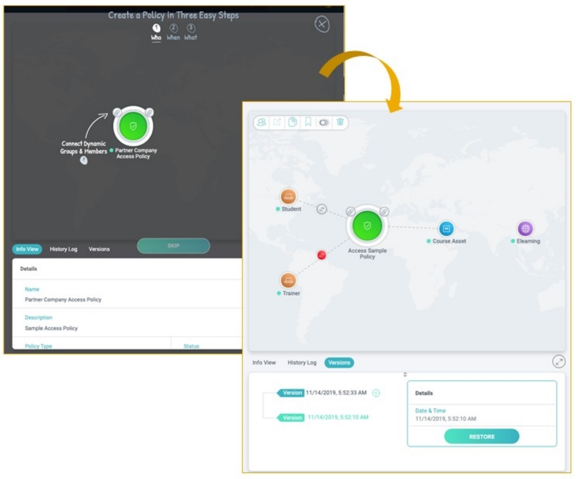
Settings	Name (Required)	Operator (Required)	Values (Required)
Apply to all	originating_ip	Equals	10.10.10.1

Organization Management Settings: demo1 2287792

Organization usage settings: Hierarchy

Figure 356: Policy Building Blocks V: Conditions (Optional)

- Use all the previous elements to build your **policy**
- Use **walkthrough assistance** to guide you through the steps to create a policy – *who*, *when,* and *what*
- When finished, don't forget to **deploy** your policy
- Use **versions** to control your policy changes



Details

Name: Partner Company Access Policy

Description: Sample Access Policy

Policy Type | Status

Version 11/14/2019, 5:52:33 AM

Version 11/14/2019, 5:52:10 AM

Details — Date & Time: 11/14/2019, 5:52:10 AM

Figure 357: Building a Policy

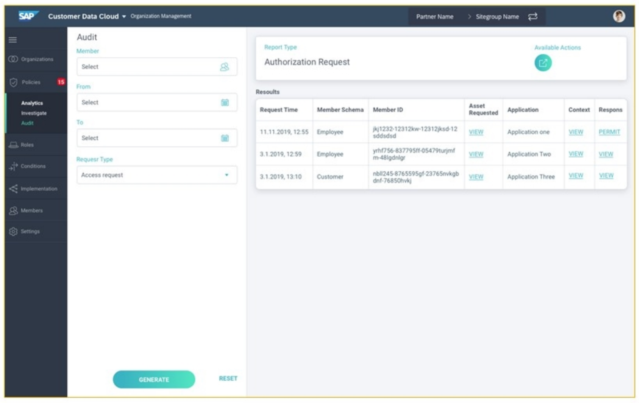
- An **authorization request** RESTful API is available
- There are two types of authorization requests:
  1. **Authorization request (basic/detailed)**: Use when you would like to do a specific authorization check on if the user has access to a specific asset
  2. **Authorization token**: Use when you would like to get a full access request of a specific identity

  https://{env-url}/runtime/{API-key}/authorization/{basic|detailed|token}/{application-client-id}
- For all end-points, a **bearer token** is required, which needs to be sent as a JWT
- Request-building parameters:
  - Bearer token, repository API key, application-client ID, user ID, organization ID, tenant ID, partner ID, site group ID

Figure 358: Authorization Requests

Authorization Request RESTful API - https://developers.gigya.com/display/GD/Organization+Management+Detailed+User+Guide#OrganizationManagementDetailedUserGuide-VI.AuthorizationRequest

- All authorization requests are audited for security reasons
- The delegated admin can generate audit reports from the Organization Management console



Audit

Member: Select

From: Select

To: Select

Requesr Type: Access request

Report Type: Authorization Request

Available Actions

Resoults

Request Time	Member Schema	Member ID	Asset Requested	Application	Context	Respons
11.11.2019, 12:55	Employee	jkj1232-12312kw-12312jksd-12 sddsdsd	VIEW	Application one	VIEW	PERMIT
3.1.2019, 12:59	Employee	yrhf756-837795ff-05479turjmf m-48lgdnlgr	VIEW	Application Two	VIEW	VIEW
3.1.2019, 13:10	Customer	nbll245-8765595gf-23765nvkgb dnf-76850hvkj	VIEW	Application Three	VIEW	VIEW

Figure 359: Policy Audit Log

LESSON SUMMARY

You should now be able to:

- Use CIAM for B2B

UNIT 13 Best Practices

UNIT OBJECTIVES

- Understand best practices for SAP Customer Data Cloud

Unit 13
Lesson 1

Understanding Best Practices

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Understand best practices for SAP Customer Data Cloud

Best Practices

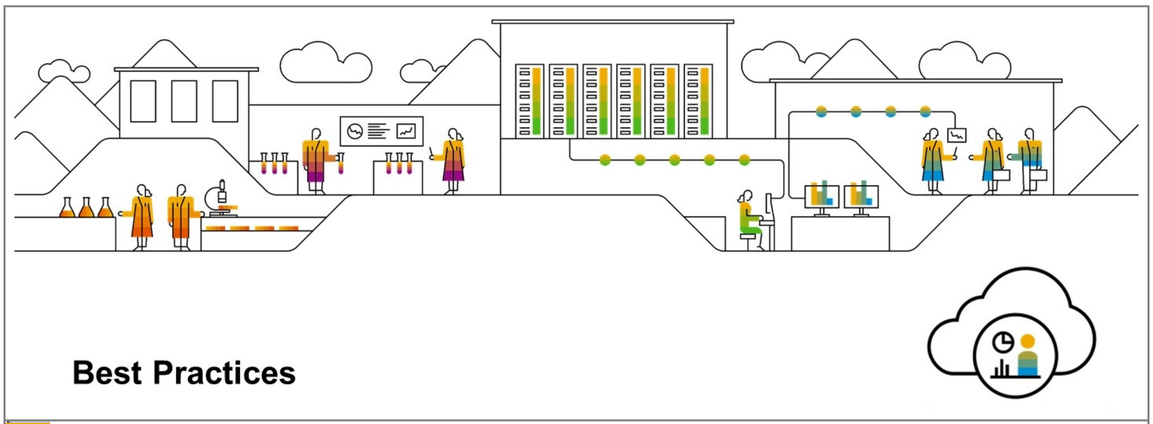
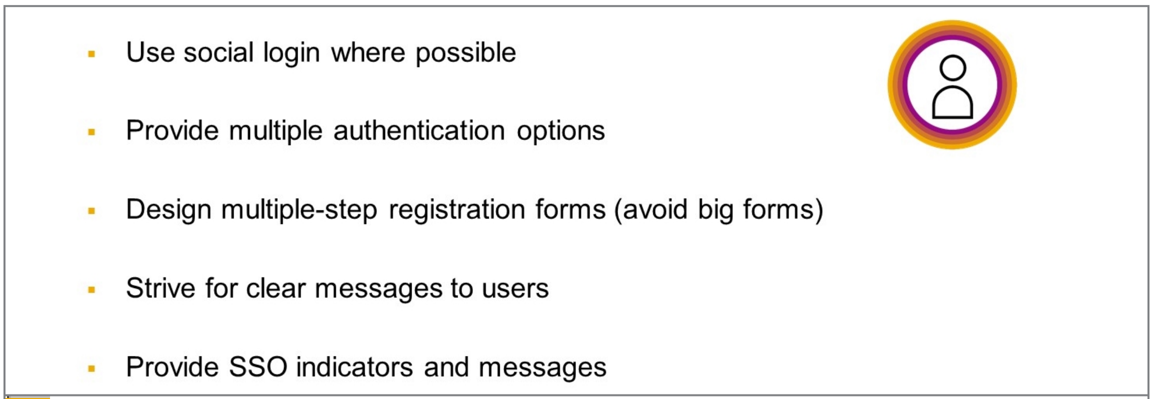


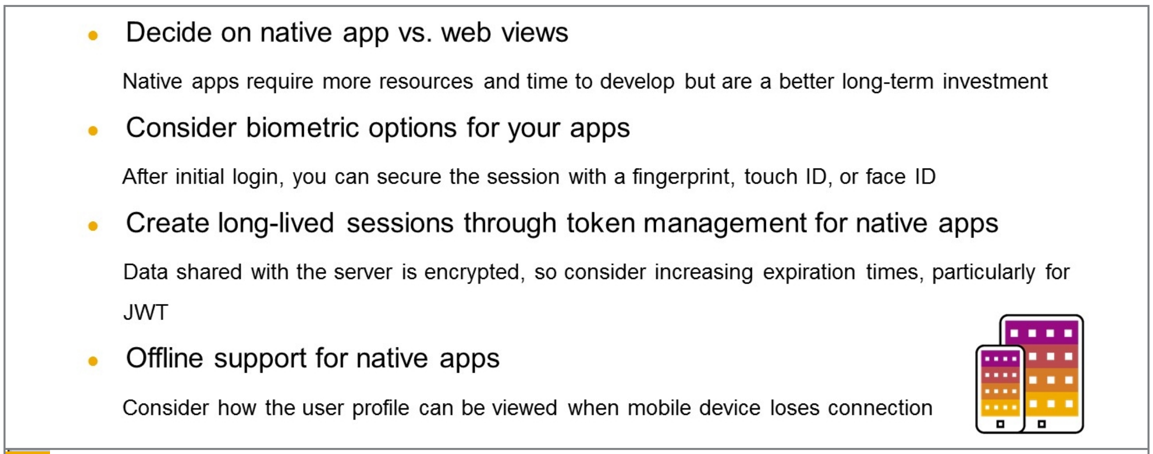
Figure 360: Best Practices



- Use social login where possible
- Provide multiple authentication options
- Design multiple-step registration forms (avoid big forms)
- Strive for clear messages to users
- Provide SSO indicators and messages

Figure 361: UX Best Practices

- Prioritize social login
- Offer different login options
- Avoid the need for scrolling
- Use conditional, progressive profiling



- Decide on native app vs. web views

  Native apps require more resources and time to develop but are a better long-term investment
- Consider biometric options for your apps

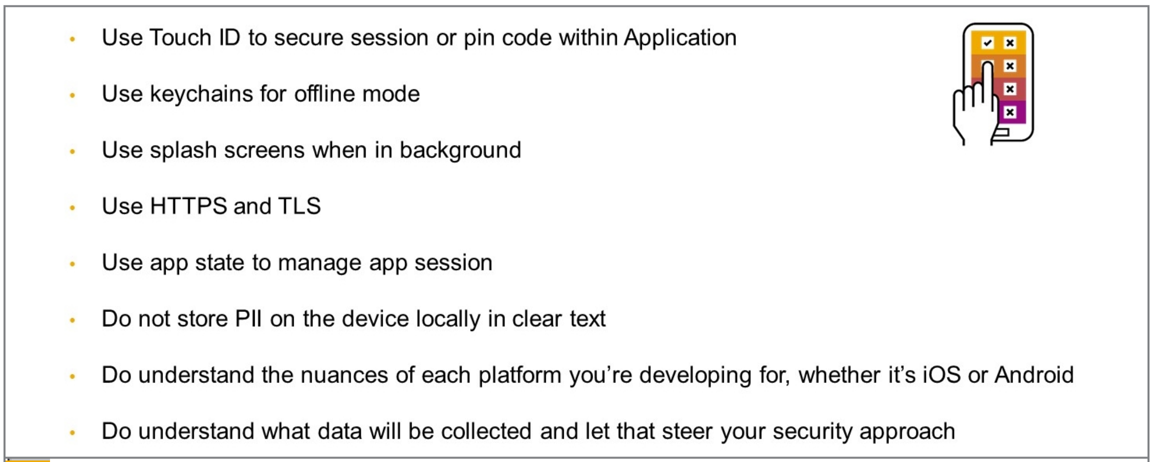
  After initial login, you can secure the session with a fingerprint, touch ID, or face ID
- Create long-lived sessions through token management for native apps

  Data shared with the server is encrypted, so consider increasing expiration times, particularly for JWT
- Offline support for native apps

  Consider how the user profile can be viewed when mobile device loses connection

Figure 362: Mobile Apps Best Practices

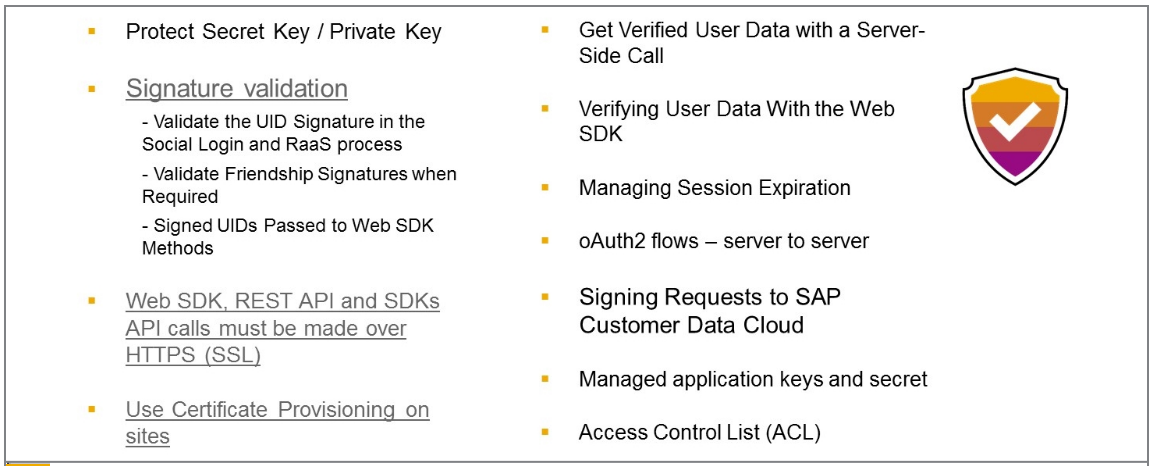
- If you go native, think about applications and resources you already have. When do you want to go live (web views are faster to implement but native will have a better UX)?
- Biometric data includes fingerprints, face ID, and touch ID. The user logs in one time, which creates a session. That session is secured with biometrics.
- Create long-lived sessions (parameter value -2 never expires). We never transmit the client session secret that is securely stored locally within the client, so the content request we sent is encrypted each at any time... changing small expiration times (JWT). This is completely different to web apps because session timeout doesn't exist.
- How to view profile when the connection goes down or the user is offline? You have to create custom special views (using local cached data, trying to (auto)reconnect view, etc...) in contrast with displaying the server not found / connection error messages from the web view (acting like a browser).
- Never use the application secret (Console/Admin/applications) in your mobile app only the application key (like it's done in the html screen-sets)



- Use Touch ID to secure session or pin code within Application
- Use keychains for offline mode
- Use splash screens when in background
- Use HTTPS and TLS
- Use app state to manage app session
- Do not store PII on the device locally in clear text
- Do understand the nuances of each platform you're developing for, whether it's iOS or Android
- Do understand what data will be collected and let that steer your security approach

Figure 363: Mobile Security Best Practices

PII - Personally Identifiable Information



- Protect Secret Key / Private Key
- Signature validation
  - Validate the UID Signature in the Social Login and RaaS process
  - Validate Friendship Signatures when Required
  - Signed UIDs Passed to Web SDK Methods
- Web SDK, REST API and SDKs API calls must be made over HTTPS (SSL)
- Use Certificate Provisioning on sites
- Get Verified User Data with a Server-Side Call
- Verifying User Data With the Web SDK
- Managing Session Expiration
- oAuth2 flows – server to server
- Signing Requests to SAP Customer Data Cloud
- Managed application keys and secret
- Access Control List (ACL)

Figure 364: Security Best Practices

Some REST APIs may function without these authorization parameters, however, when that occurs, these calls are treated as client-side calls and all client-side rate limits will apply. In order to not reach client-side rate limits that may impact your implementation when using server-to-server REST calls, it is Recommended Best Practice to always sign the request or use a secret.

If this parameter is not passed, the default TTL of the returned JWT is 300 seconds.

https://developers.gigya.com/display/GD/OAuth+2.0+Compliant+REST+API

When using an HTTPS domain, when loading the Gigya Web SDK on a web page, load it from our HTTPS domain (i.e., reference "CDNS" and not "CDN", as per the example below). This will not only load the SDK itself over HTTPS but will cause the SDK to perform all its communications with the Gigya server over HTTPS as well.

Advanced Cookie Reference

https://developers.gigya.com/display/GD/Advanced+Cookie+Reference

Managing Session Expiration

https://developers.gigya.com/display/GD/Managing+Session+Expiration

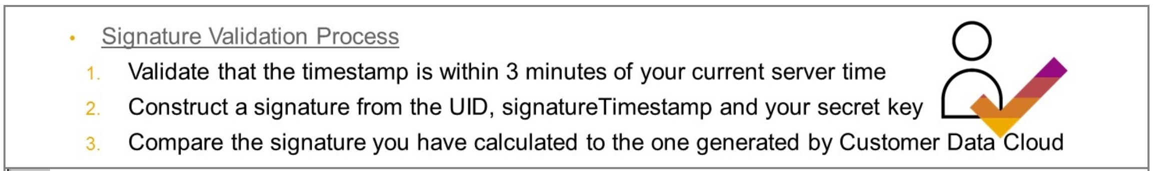
JWT expiration

https://developers.gigya.com/display/GD/accounts.getJWT+JS

Signing Requests to SAP Customer Data Cloud

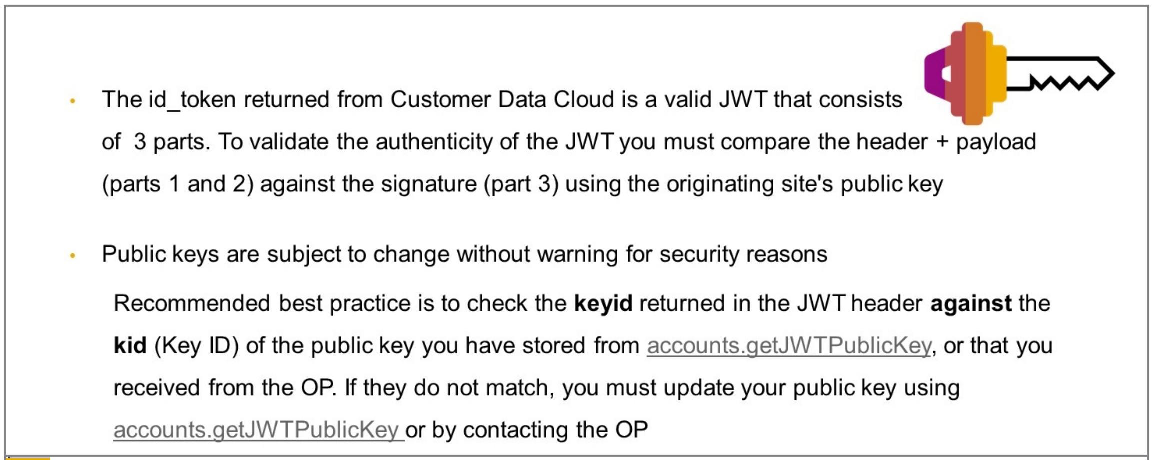
https://developers.gigya.com/display/GD/Signing+Requests+to+SAP+Customer+Data+Cloud

https://developers.gigya.com/display/GD/Security+Best+Practices



- Signature Validation Process
  1. Validate that the timestamp is within 3 minutes of your current server time
  2. Construct a signature from the UID, signatureTimestamp and your secret key
  3. Compare the signature you have calculated to the one generated by Customer Data Cloud

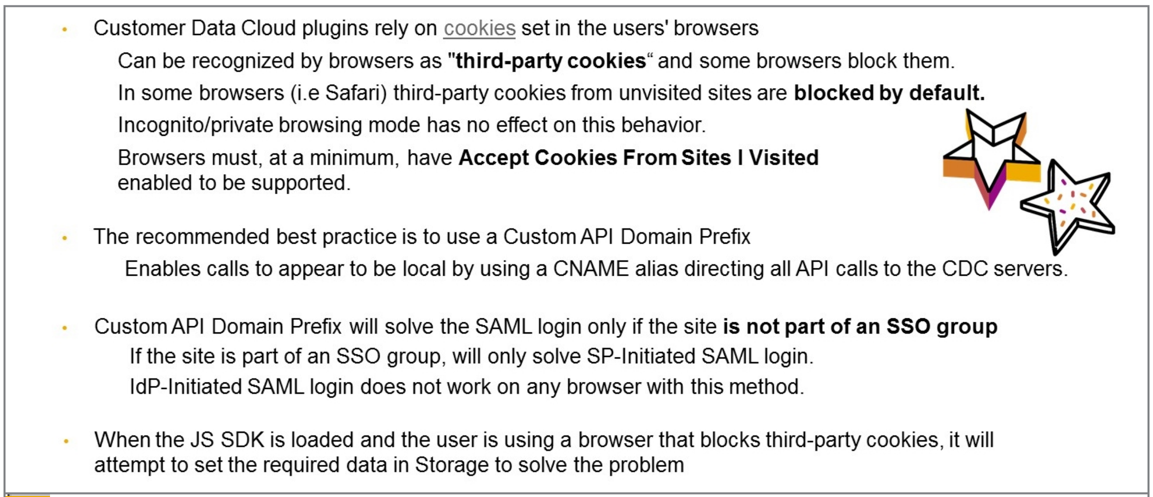
Figure 365: Signature Validation Best Practices



- The id_token returned from Customer Data Cloud is a valid JWT that consists of 3 parts. To validate the authenticity of the JWT you must compare the header + payload (parts 1 and 2) against the signature (part 3) using the originating site's public key
- Public keys are subject to change without warning for security reasons

  Recommended best practice is to check the **keyid** returned in the JWT header **against** the **kid** (Key ID) of the public key you have stored from accounts.getJWTPublicKey, or that you received from the OP. If they do not match, you must update your public key using accounts.getJWTPublicKey or by contacting the OP

Figure 366: ID_Token Validation Best Practices



- Customer Data Cloud plugins rely on cookies set in the users' browsers

  Can be recognized by browsers as "**third-party cookies**“ and some browsers block them.
  In some browsers (i.e Safari) third-party cookies from unvisited sites are **blocked by default.**
  Incognito/private browsing mode has no effect on this behavior.
  Browsers must, at a minimum, have **Accept Cookies From Sites I Visited** enabled to be supported.
- The recommended best practice is to use a Custom API Domain Prefix

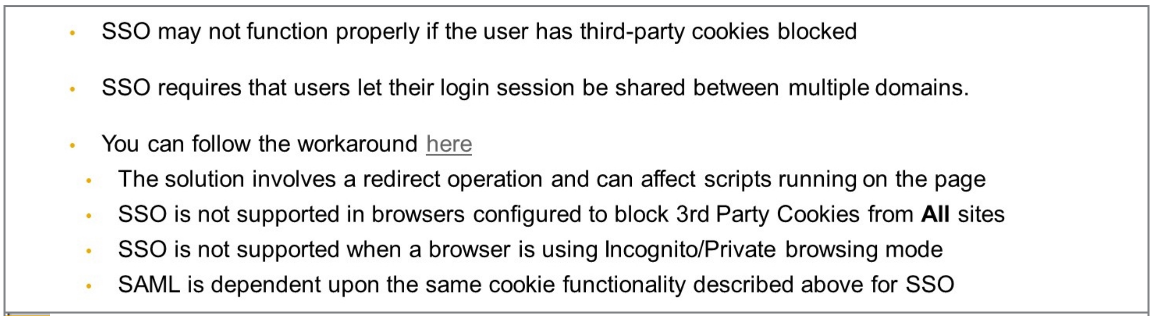
  Enables calls to appear to be local by using a CNAME alias directing all API calls to the CDC servers.
- Custom API Domain Prefix will solve the SAML login only if the site **is not part of an SSO group**

  If the site is part of an SSO group, will only solve SP-Initiated SAML login.
  IdP-Initiated SAML login does not work on any browser with this method.
- When the JS SDK is loaded and the user is using a browser that blocks third-party cookies, it will attempt to set the required data in Storage to solve the problem

Figure 367: Cookie Best Practices - Blocked 3rd Party Cookies

https://developers.gigya.com/display/GD/Blocked+Third-Party+Cookies

By default, when the Gigya JS SDK is loaded in a webpage, if it sees the user is using a browser that blocks third-party cookies by default (e.g., Safari), it will attempt to set the required data in Local Storage to solve the problem.



- SSO may not function properly if the user has third-party cookies blocked
- SSO requires that users let their login session be shared between multiple domains.
- You can follow the workaround here
  - The solution involves a redirect operation and can affect scripts running on the page
  - SSO is not supported in browsers configured to block 3rd Party Cookies from **All** sites
  - SSO is not supported when a browser is using Incognito/Private browsing mode
  - SAML is dependent upon the same cookie functionality described above for SSO

Figure 368: Supporting SSO & SAML Login in Browsers that Block Third-Party Cookies

https://developers.gigya.com/display/GD/Blocked+Third-Party+Cookies

OAuth2-compliant REST API

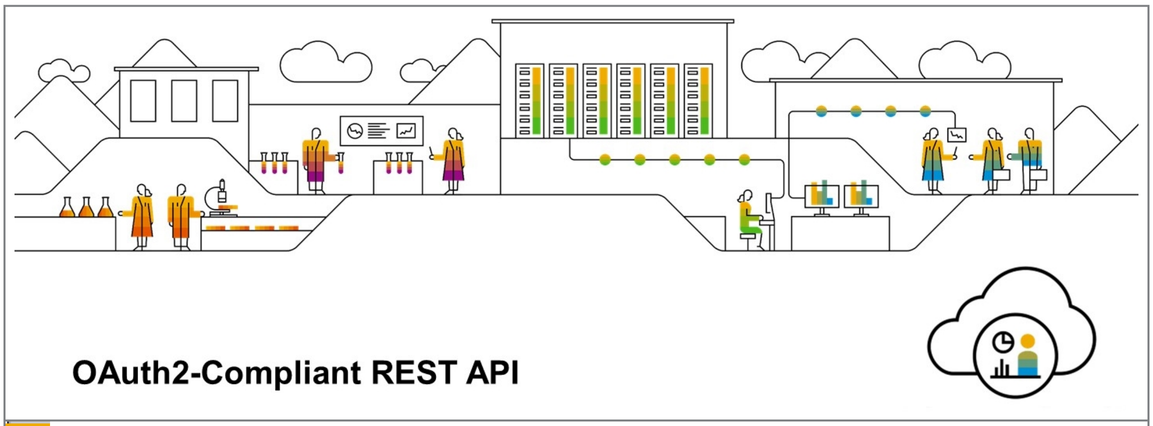
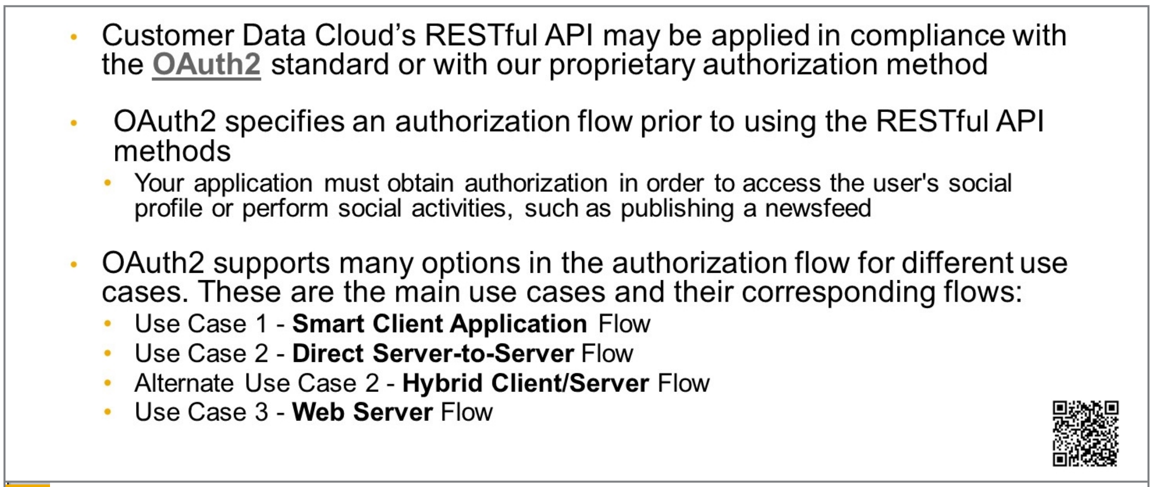


Figure 369: OAuth2-compliant REST API

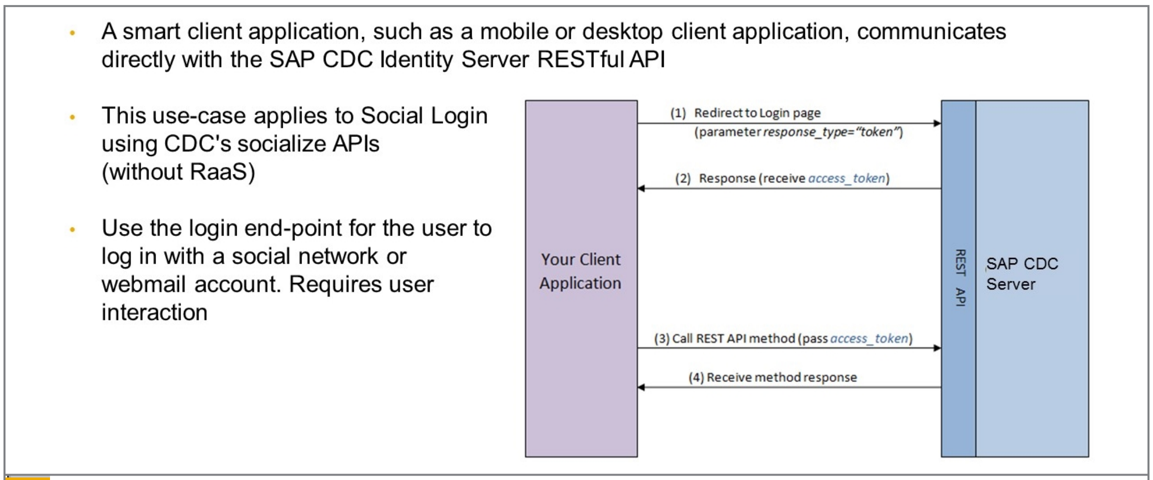


- Customer Data Cloud's RESTful API may be applied in compliance with the OAuth2 standard or with our proprietary authorization method
- OAuth2 specifies an authorization flow prior to using the RESTful API methods
  - Your application must obtain authorization in order to access the user's social profile or perform social activities, such as publishing a newsfeed
- OAuth2 supports many options in the authorization flow for different use cases. These are the main use cases and their corresponding flows:
  - Use Case 1 - **Smart Client Application** Flow
  - Use Case 2 - **Direct Server-to-Server** Flow
  - Alternate Use Case 2 - **Hybrid Client/Server** Flow
  - Use Case 3 - **Web Server** Flow

Figure 370: OAuth2 Compliant RESTful API

http://wiki.oauth.net/OAuth-2

https://developers.gigya.com/display/GD/OAuth+2.0+Compliant+REST+API



- A smart client application, such as a mobile or desktop client application, communicates directly with the SAP CDC Identity Server RESTful API
- This use-case applies to Social Login using CDC's socialize APIs (without RaaS)
- Use the login end-point for the user to log in with a social network or webmail account. Requires user interaction

Your Client Application
(1) Redirect to Login page (parameter response_type="token")
(2) Response (receive access_token)
(3) Call REST API method (pass access_token)
(4) Receive method response
REST API
SAP CDC Server

Figure 371: Use Case 1 - Smart Client Application Flow

Your Client Application:

1. Log the user in via the CDC service by redirecting the user to the CDC login end-point URL. Pass the response_type="token" parameter.
2. In the Login response you will receive the access_token. This token is your authorization ticket for accessing the current user's social profile.
3. Pass the access_token with each RESTful API method call associated with the current user.
4. Receive the RESTful API method response.

https://developers.gigya.com/display/GD/OAuth+2.0+Compliant+REST+API

- In this use case, the RESTful API calls are not bound to an active session
  - For methods which must be associated to a specific user (such as setStatus, getFriendsInfo, etc.), retrieve the user's UID from your database and pass it as a parameter
  - For methods which are not associated with a user (such as reports.getSocializeStats), there is no need to pass a UID parameter

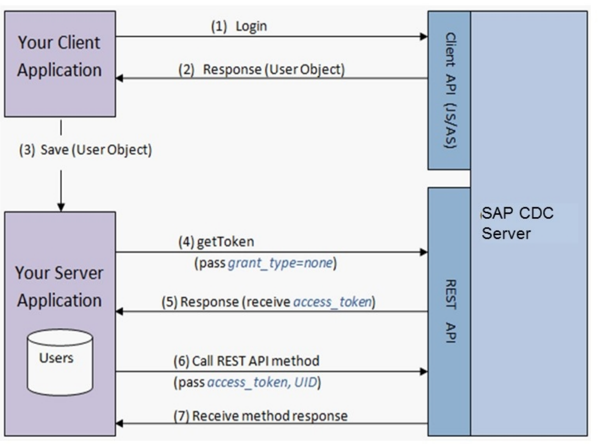

Your Client Application
(1) Login
(2) Response (User Object)
Client API (JS/AS)
(3) Save (User Object)
SAP CDC Server
Your Server Application
Users
(4) getToken (pass grant_type=none)
(5) Response (receive access_token)
(6) Call REST API method (pass access_token, UID)
(7) Receive method response
REST API

Figure 372: Use Case 2 - Direct Server-to-Server Flow

This use case applies when using CDC's Registration-as-a-Service (RaaS).

Your Client Application:

1. Log in the user via the CDC service using the web SDK method calls, socialize.login or socialize.notifyLogin. You may also use the ready-made login add-on, socialize.showLoginUI.
2. In the login response, you will receive the user object.
3. Pass the user object to your server application.

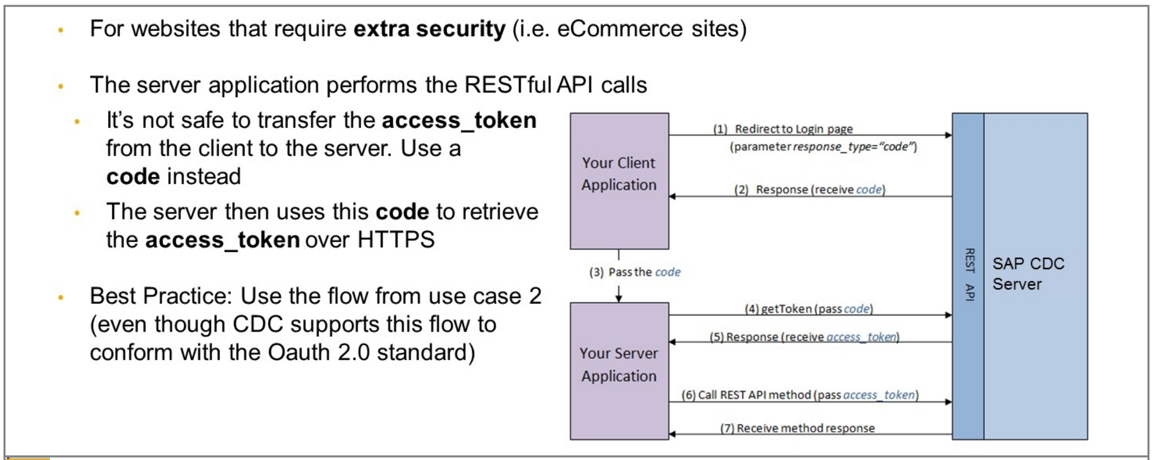
Your Server Application:

1. Obtain the access_token, using the getToken end-point, and pass the grant_type="none" parameter. The getToken method must be called over HTTPS.
2. Receive the getToken method response, which will contain the access_token. Note that in this use case, the token is not related to a specific user.
3. Pass the access_token with each RESTful API method call. If the method call is associated with a specific user, also pass the user's UID as a parameter.
4. Receive RESTful API method response.

Alternate Use Case 2 - Hybrid Client/Server Flow

As above, in this case you use both the client and server applications to communicate with CDC's RESTful and Client API's, but in reverse order. First the server requests an access token, then that token is set as a WebSDK Configuration object when loading the CDC Web SDK.

https://developers.gigya.com/display/GD/OAuth+2.0+Compliant+REST+API



- For websites that require **extra security** (i.e. eCommerce sites)
- The server application performs the RESTful API calls
  - It's not safe to transfer the **access_token** from the client to the server. Use a **code** instead
  - The server then uses this **code** to retrieve the **access_token** over HTTPS
- Best Practice: Use the flow from use case 2 (even though CDC supports this flow to conform with the Oauth 2.0 standard)

Your Client Application
(1) Redirect to Login page (parameter *response_type="code"*)
(2) Response (receive *code*)
(3) Pass the *code*
Your Server Application
(4) getToken (pass *code*)
(5) Response (receive *access_token*)
(6) Call REST API method (pass *access_token*)
(7) Receive method response
REST API
SAP CDC Server

Figure 373: Use Case 3 - Web Server Flow

This use case applies to Social Login using CDC's socialize APIs (without RaaS).

This authorization code flow is best used in web and mobile apps. Since the authorization code grant has the extra step of exchanging the authorization code for the access token, it provides an additional layer of security not present in the implicit grant type. The code exchange step ensures that an attacker isn't able to intercept the access token, since the access token is always sent via a secure backchannel between the application and the OAuth server.

Use case 2 is recommended over this one, because in use case 2's server step 1, the access token is always obtained using HTTPS.

Use case 3 is more secure than use case 1, because with use case 1, anyone using the browser can see and use the access token. Use case 3's extra authorization code step prevents this by authorizing the client before using the access token.

Your Client Application:

1. Log the user in via the CDC service's Login Plugin or the socialize.login API method of the CDC Web SDK. Important: Set the authCodeOnly parameter to true.
2. In the onLogin event data (or in the socialize.login API method response) you will receive the authCode.
3. Pass the authCode to your server application.(Alternatively, you may redirect the user to the CDC login end-point URL. Pass the response_type="code" parameter. Receive the code in the response, and pass it to the server. Read more in the Login end-point section below.)

Your Server Application:

1. Obtain the access_token using the getToken end-point. Pass the code and grant_type="authorization_code" as parameters. The getToken method must be called over HTTPS.
2. Receive the getToken method response. The response will contain the access_token. This token is the authorization ticket for accessing the current user's social profile.
3. Pass the access_token with each RESTful API method call associated to that user.
4. Receive the RESTful API method response.

https://developers.gigya.com/display/GD/OAuth+2.0+Compliant+REST+API

https://developer.okta.com/blog/2018/04/10/oauth-authorization-code-grant-type

API Rate Limits

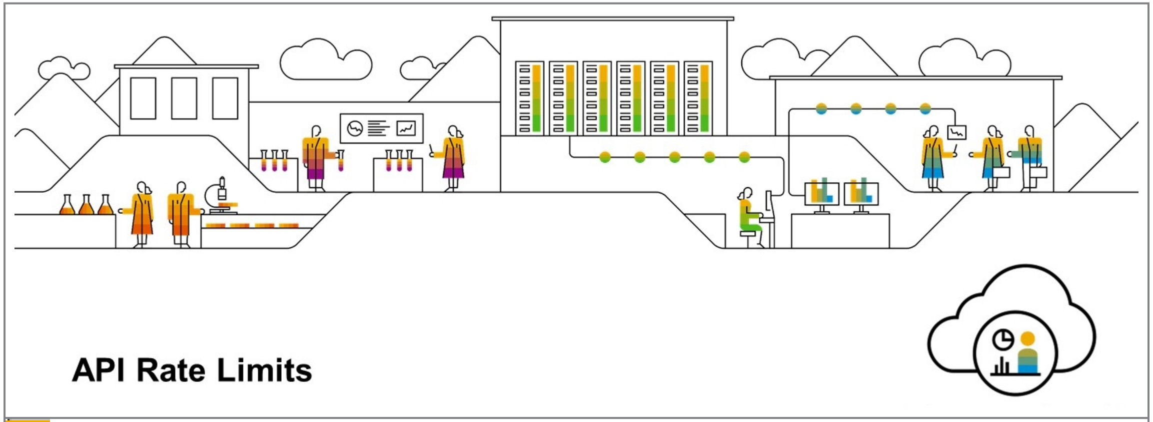
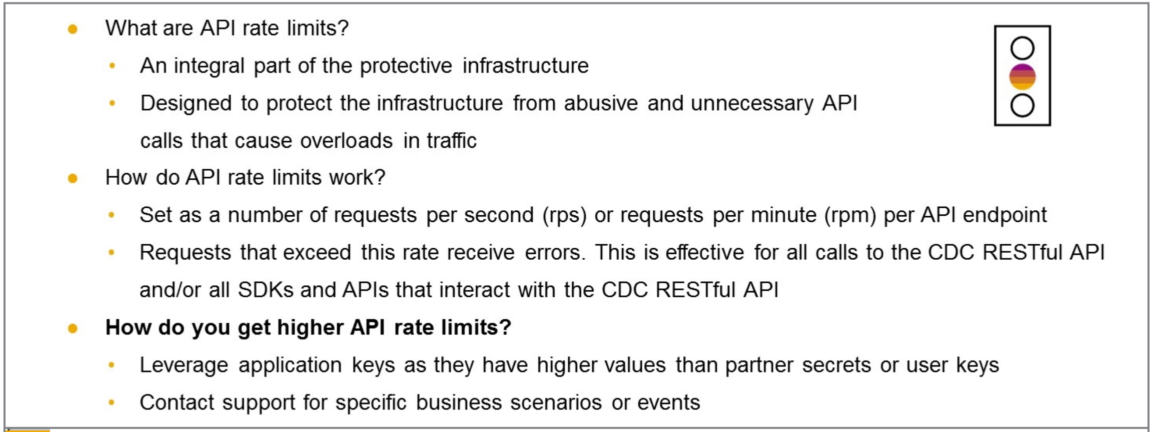


Figure 374: API Rate Limits



- What are API rate limits?
  - An integral part of the protective infrastructure
  - Designed to protect the infrastructure from abusive and unnecessary API calls that cause overloads in traffic
- How do API rate limits work?
  - Set as a number of requests per second (rps) or requests per minute (rpm) per API endpoint
  - Requests that exceed this rate receive errors. This is effective for all calls to the CDC RESTful API and/or all SDKs and APIs that interact with the CDC RESTful API
- **How do you get higher API rate limits?**
  - Leverage application keys as they have higher values than partner secrets or user keys
  - Contact support for specific business scenarios or events

Figure 375: API Rate Limits

For security reasons, CDC does not publish other information regarding API rate limits.

- Usually, API rate limits are set to 50 rps per endpoint except for login, getschema, getaccount, and few others which allow 300-500 rps.
- The rate for client-side calls is much lower than server-side at about 60 requests per minute (rpm)... be careful with mobile apps!
- Partner, user, and app each set different limits.

Each customer has a default set for API rate limits

- These limits are not enough for Black Friday or peak times, but usually are fine for typical days
- For big events, contact support two weeks in advance to set up special rate limits.
- Acceptable Use Policy: https://developers.gigya.com/display/GD/Acceptable+Use +Policy

User Imports

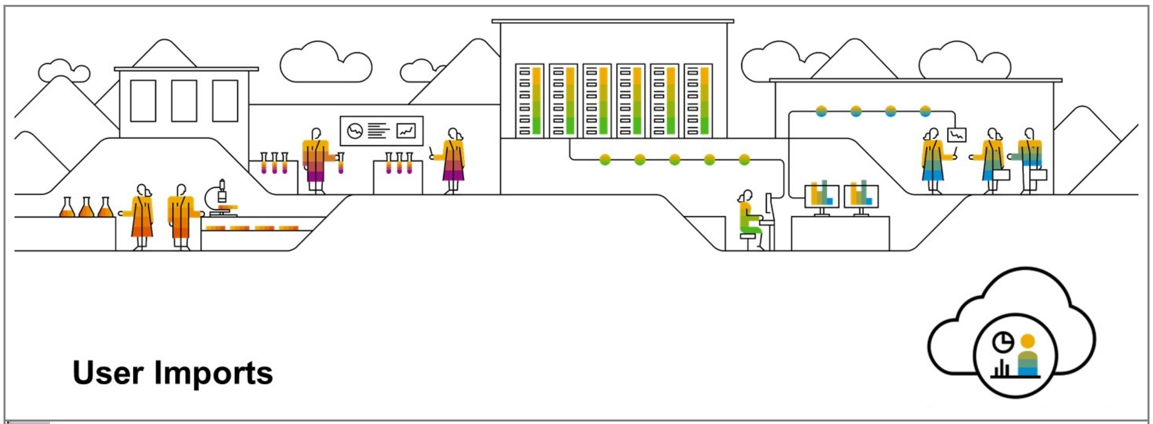
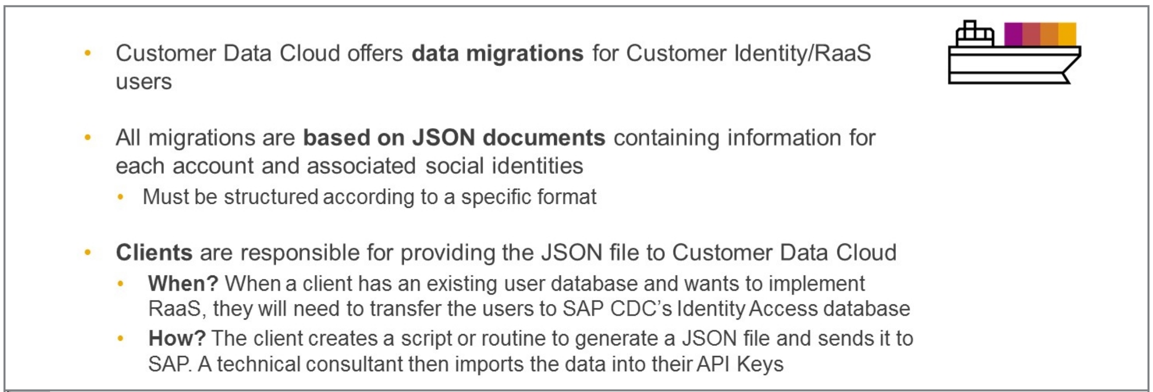


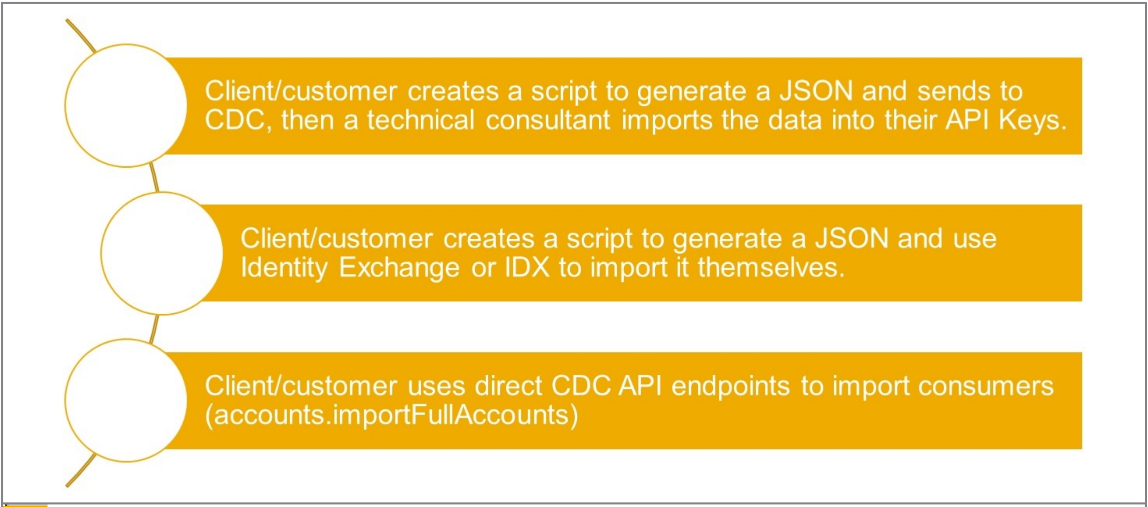
Figure 376: User Imports



- Customer Data Cloud offers **data migrations** for Customer Identity/RaaS users
- All migrations are **based on JSON documents** containing information for each account and associated social identities
  - Must be structured according to a specific format
- **Clients** are responsible for providing the JSON file to Customer Data Cloud
  - **When?** When a client has an existing user database and wants to implement RaaS, they will need to transfer the users to SAP CDC's Identity Access database
  - **How?** The client creates a script or routine to generate a JSON file and sends it to SAP. A technical consultant then imports the data into their API Keys

Figure 377: User Migrations

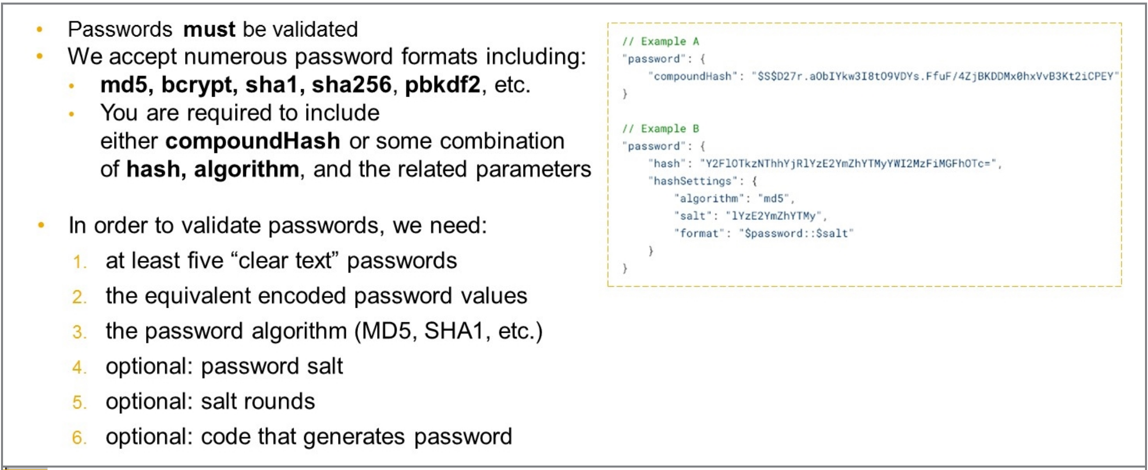
- https://developers.gigya.com/display/GD/Import+Guides
- https://developers.gigya.com/display/GD/Users
- Check that the hashing algorithm for the password is compatible with CDC.
  - The hash algorithm is used to encrypt the password
  - The supported hash algorithms are: md5, sha1, sha256, md5_crypt, bcrypt, pbkdf2 and drupal
- See important details about the password in the help documentation here: https://developers.gigya.com/display/GD/accounts.importFullAccount+REST


Client/customer creates a script to generate a JSON and sends to CDC, then a technical consultant imports the data into their API Keys.

Client/customer creates a script to generate a JSON and use Identity Exchange or IDX to import it themselves.

Client/customer uses direct CDC API endpoints to import consumers (accounts.importFullAccounts)

Figure 378: User Migration Options

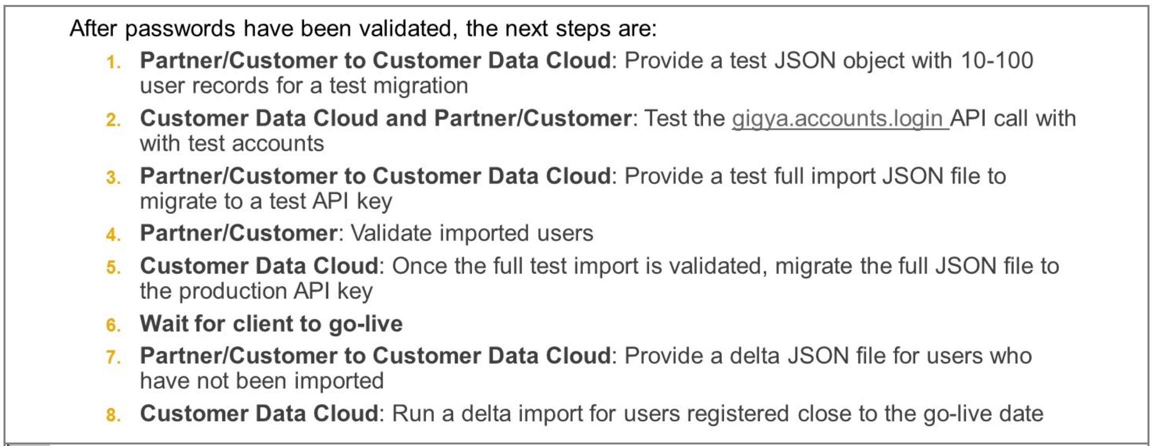

- Passwords **must** be validated
- We accept numerous password formats including:
  - **md5, bcrypt, sha1, sha256**, **pbkdf2**, etc.
  - You are required to include either **compoundHash** or some combination of **hash, algorithm**, and the related parameters
- In order to validate passwords, we need:
  1. at least five “clear text” passwords
  2. the equivalent encoded password values
  3. the password algorithm (MD5, SHA1, etc.)
  4. optional: password salt
  5. optional: salt rounds
  6. optional: code that generates password

```
// Example A
"password": {
    "compoundHash": "$S$D27r.aObIYkw3I8tO9VDYs.FfuF/4ZjBKDDMx0hxVvB3Kt2iCPEY"
}

// Example B
"password": {
    "hash": "Y2FlOTkzNThhYjRlYzE2YmZhYTMyYWI2MzFiMGFhOTc=",
    "hashSettings": {
        "algorithm": "md5",
        "salt": "lYzE2YmZhYTMy",
        "format": "$password::$salt"
    }
}
```

Figure 379: Validate Passwords

Using proprietary/legacy password hashing algoritms:

1. User submits login/password to CDC
2. Proprietary/Legacy hashing of user-submitted password
3. If the hashed password from step above matches the stored hashed legacy password:
4. hashes the clear-text user-submitted password using one of the standard supported by CDC hashing algorithms
5. stores the new hashed password



After passwords have been validated, the next steps are:

1. **Partner/Customer to Customer Data Cloud**: Provide a test JSON object with 10-100 user records for a test migration
2. **Customer Data Cloud and Partner/Customer**: Test the gigya.accounts.login API call with with test accounts
3. **Partner/Customer to Customer Data Cloud**: Provide a test full import JSON file to migrate to a test API key
4. **Partner/Customer**: Validate imported users
5. **Customer Data Cloud**: Once the full test import is validated, migrate the full JSON file to the production API key
6. **Wait for client to go-live**
7. **Partner/Customer to Customer Data Cloud**: Provide a delta JSON file for users who have not been imported
8. **Customer Data Cloud**: Run a delta import for users registered close to the go-live date

Figure 380: User Migration Process

https://developers.gigya.com/display/GD/accounts.login+JS

More information here: https://www.sap.com/cxworks/article/435028288/User_Migration_Playbook_or_Template_Download

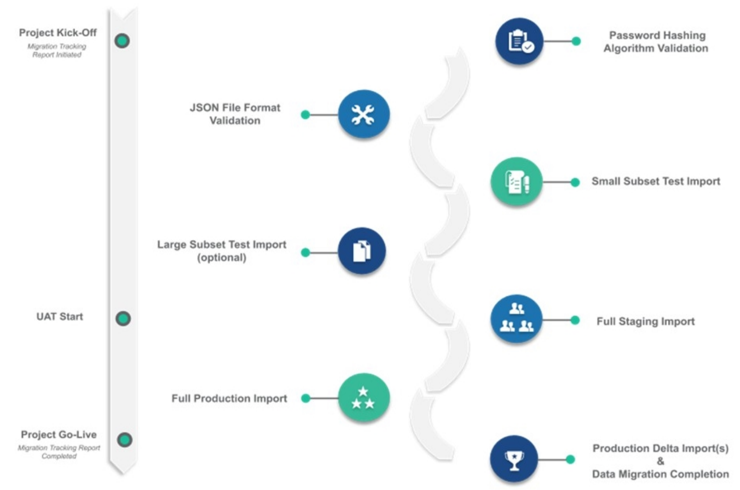

Project Kick-Off
Migration Tracking Report Initiated
UAT Start
Project Go-Live
Migration Tracking Report Completed
Password Hashing Algorithm Validation
JSON File Format Validation
Small Subset Test Import
Large Subset Test Import (optional)
Full Staging Import
Full Production Import
Production Delta Import(s) & Data Migration Completion

Figure 381: Data Migration Overview

De-duplication scenarios:

- The de-duplication will be based on business logic defined by you, as really in this case there isn't any one-size fits all solution, for example you could decide that an account already exists in CDC if the email address is already taken (that is assuming you do not have duplicate email addresses in Site N yourselves), or your could push the logic further by checking for other profile attributes. As a minimum, the opt-ins are typically imported for users of additional brands.
- As for the decision to update existing CDC accounts with Site N data or not, again this will highly depend on your use case. For example you may have a Site N with a more highly engaged community than your already existing CDC sites, in which case it might make sense to update the password in CDC with the password from Site N.
- Because of this wide range of possibilities, the development of the deduplication script is usually best left in the hands of your own technical teams, but your CDC Technical

Consultant will be more than happy to guide you through the use of CDC's APIs necessary to this development.

Key Resources

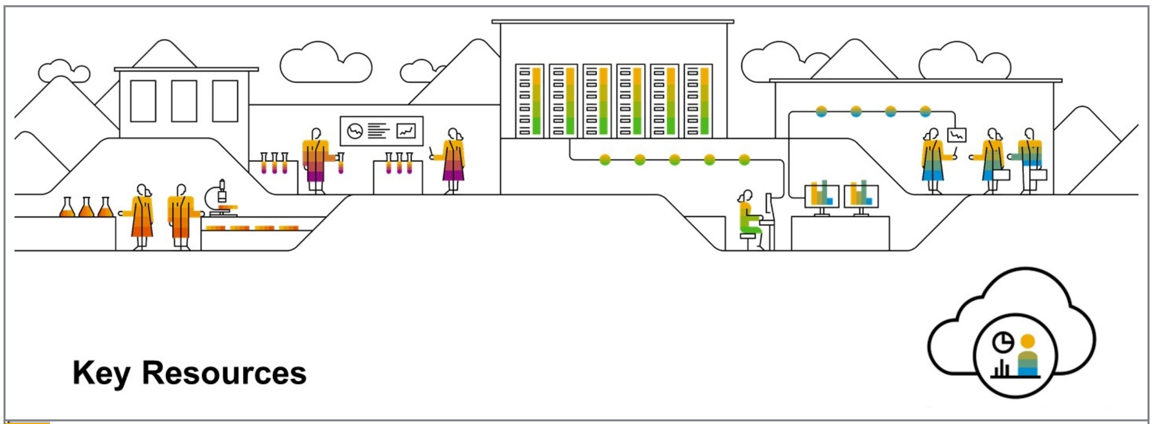
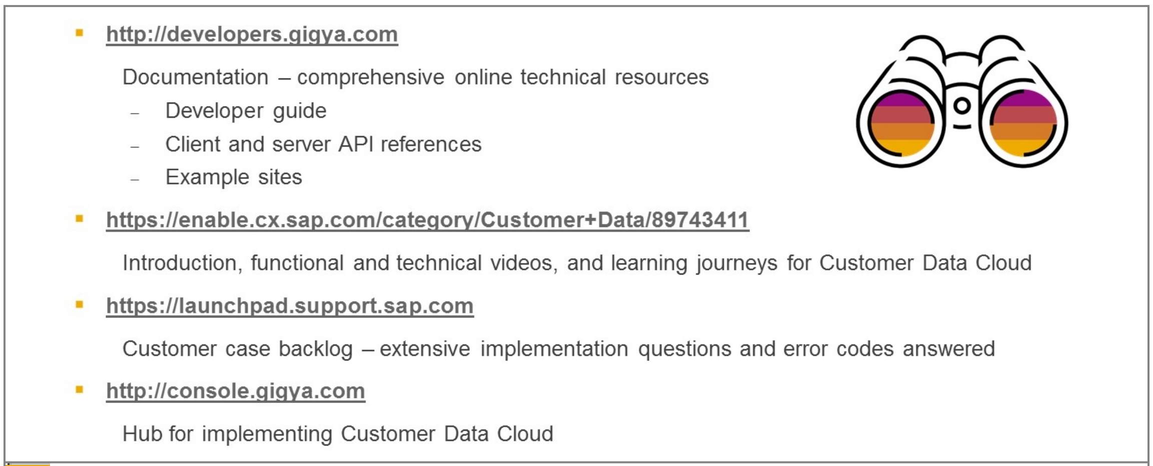


Figure 382: Key Resources



- http://developers.gigya.com

  Documentation – comprehensive online technical resources
  - Developer guide
  - Client and server API references
  - Example sites
- https://enable.cx.sap.com/category/Customer+Data/89743411

  Introduction, functional and technical videos, and learning journeys for Customer Data Cloud
- https://launchpad.support.sap.com

  Customer case backlog – extensive implementation questions and error codes answered
- http://console.gigya.com

  Hub for implementing Customer Data Cloud

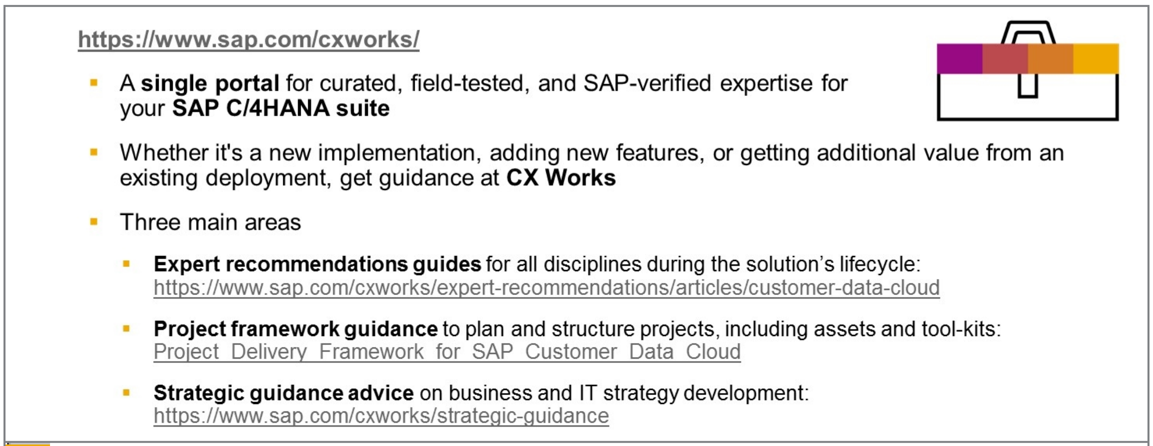
Figure 383: Customer Data Cloud Developer Resources

http://developers.gigya.com

https://enable.cx.sap.com/category/Customer+Data/89743411

https://launchpad.support.sap.com

http://console.gigya.com



https://www.sap.com/cxworks/

- A **single portal** for curated, field-tested, and SAP-verified expertise for your **SAP C/4HANA suite**
- Whether it's a new implementation, adding new features, or getting additional value from an existing deployment, get guidance at **CX Works**
- Three main areas
  - **Expert recommendations guides** for all disciplines during the solution's lifecycle: https://www.sap.com/cxworks/expert-recommendations/articles/customer-data-cloud
  - **Project framework guidance** to plan and structure projects, including assets and tool-kits: Project_Delivery_Framework_for_SAP_Customer_Data_Cloud
  - **Strategic guidance advice** on business and IT strategy development: https://www.sap.com/cxworks/strategic-guidance

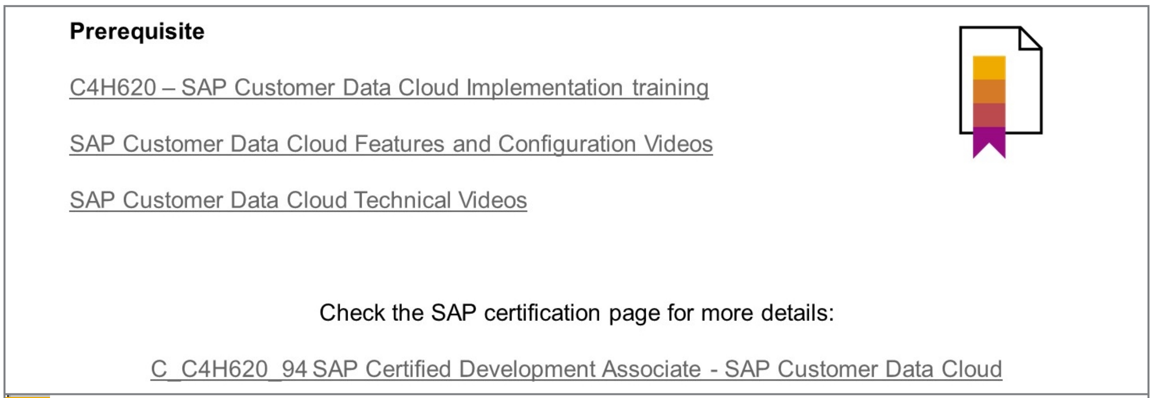
Figure 384: CX Works

https://www.sap.com/cxworks/

https://www.sap.com/cxworks/expert-recommendations/articles/customer-data-cloud

https://www.sap.com/cxworks/article/432583102/
Project_Delivery_Framework_for_SAP_Customer_Data_Cloud

https://www.sap.com/cxworks/strategic-guidance



Prerequisite

C4H620 – SAP Customer Data Cloud Implementation training

SAP Customer Data Cloud Features and Configuration Videos

SAP Customer Data Cloud Technical Videos

Check the SAP certification page for more details:

C_C4H620_94 SAP Certified Development Associate - SAP Customer Data Cloud

Figure 385: SAP Certified Development Associate SAP CDC Exam

https://training.sap.com/trainingpath/Cloud-SAP+Customer+Experience+%28C4HANA
%29-Customer+Data+Cloud

https://enable.cx.sap.com/channel/Customer+Data+Cloud+Features+and+Configurations/
95027061

https://enable.cx.sap.com/channel/Customer+Data+Cloud+Technical/95020731

https://training.sap.com/trainingpath/Cloud-SAP+Customer+Experience+%28C4HANA
%29-Customer+Data+Cloud

LESSON SUMMARY

You should now be able to:

- Understand best practices for SAP Customer Data Cloud