

# Schema Editor

Registration-as-a-Service (RaaS) is a premium platform that requires separate activation. If **RaaS** is not part of your site package, please contact Gigya by filling in a [support form](#). You can access the support page by clicking **Support** on the upper menu after logging into your Gigya Console.

## Description

Gigya's Schema is an integral part of our [Customer Identity](#) (RaaS) platform which allows you to add and access up to 1000 (one-thousand) additional custom fields beyond the default [Profile object](#) for each of your sites (API keys). If you are using Gigya's [Data Store](#), you may also edit its schema using the Schema Editor. The editor gives you an un-paralleled ability to customize, not only the information you store for each user, but the unique experience every user has when visiting your site.

You can use the **Schema Editor** of the Gigya Console to easily and interactively edit your site's schema whenever the need arises, and these new fields will be ready for use immediately.

This utility is only available to Console users that have the necessary Console permissions to use at least one of the following APIs:

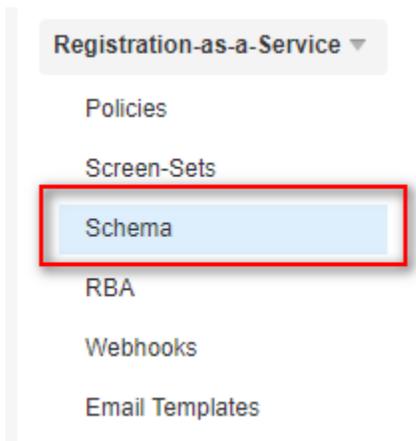
- [accounts.getSchema](#)
- [accounts.setSchema](#)

If your Console user account only has permissions for [accounts.getSchema](#), you will be able to view the current schema but all editing capabilities will be disabled.

If your account has the necessary permissions, you can locate the **Schema** page of the [Console](#) from the left-hand menu of the Dashboard.

## Using The Schema Editor

Navigate to the Schema Editor of the [Gigya Console](#).



## Dynamic Schema

When first arriving at the Schema Editor, the **data** node of your site's schema will be selected and you will see the **Enable dynamic schema** checkbox. Dynamic schema affects the APIs you can use to create new fields in your schema.

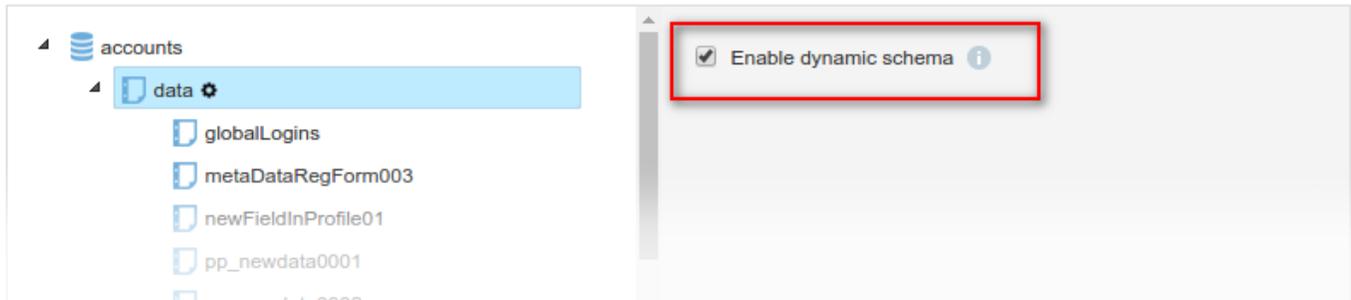
## Schema

Gigya's schema is our cloud hosted solution for storing all of your users' data. Gigya's schema can support up to 1000 (one thousand) additional custom fields beyond the default schema, and all of these fields may be customized to validate user input as necessary. Use this editor to review and change your site's current schema. For more information, please read the [Developer's Guide](#).

Data is accurate as of: **Jun 4, 2017, 10:55:49**

[Refresh](#)

[Create Data Field](#)



- When Dynamic Schema is disabled, you can only add new fields to your site's schema using the [Schema Editor](#) of the Gigya Console, the [Screen-Sets UI Builder](#) or the [accounts.setSchema](#) API method.
- When Dynamic Schema is enabled, in addition to the Schema Editor and the [accounts.setSchema](#) API, you can also use the [accounts.setAccountInfo](#) API method to add new fields to your schema.

Even when **Dynamic Schema** is enabled, it is recommended best practice to only create new fields in your schema via the **Schema Editor**, the **Screen-Sets UI Builder**, or the [accounts.setSchema](#) API method. Creating new fields using [accounts.setAccountInfo](#) automatically creates the field with its write access set to **serverOnly**, which means that the field will only accept data from the initial user who triggered the [setAccountInfo](#) call to create the field, additional users will not be able to use this field until you manually change the write access to **clientModify**. If your implementation depends on new fields created with [accounts.setAccountInfo](#), it may be adversely affected.

## Editing Existing Field Properties

When you select a field from the tree in the left-hand pane of the editor, or after clicking the **Create Data Field** button and creating a new field, a **Properties** dialog will display in the right-hand pane.

Existing fields support editing of the following properties:

- **Write Access**
- **Required**
- **Nullable**

For an explanation of the above fields, see [Adding New Fields](#), below.

## Schema

Gigya's schema is our cloud hosted solution for storing all of your users' data. Gigya's schema can support up to 1000 (one thousand) additional custom fields beyond the default schema, and all of these fields may be customized to validate user input as necessary. Use this editor to review and change your site's current schema. For more information, please read the [Developer's Guide](#).

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

The screenshot shows the Schema Editor interface. On the left, a tree view displays the 'accounts' schema structure. The 'data' node is expanded, and the 'subscribe' field is selected. On the right, a configuration panel for the 'subscribe' field is shown, enclosed in a red border. The panel includes the following fields and options:

- Name:** data.subscribe
- Type:** boolean
- Write Access:** clientModify
- Required:**
- Nullable:**

## Adding New Fields

1. Click the **Create Field** button on the top-right of the editor. Note that you can not add additional nodes beneath existing fields.

### Schema

Gigya's schema is our cloud hosted solution for storing all of your users' data. Gigya's schema can support up to 1000 (one thousand) additional custom fields beyond the default schema, and all of these fields may be customized to validate user input as necessary. Use this editor to review and change your site's current schema. For more information, please read the [Developer's Guide](#).

Data is accurate as of: **Aug 10, 2017, 10:38:01**

The screenshot shows the Schema Editor interface with the 'Create Field' window open. The tree view on the left shows the 'accounts' schema with the 'data' node expanded. The 'Create Field' window on the right has a 'Name' field containing 'data.car'.

2. In the **Create Field** window:

- a. If you have the [Data Store](#) enabled for your account, select whether to create the field in the **Accounts** or the **DS** schema.
  - b. If you have [Lite Registration](#) or [Subscription Management](#) enabled for your account, select the type: a **Data** or **Subscription** field.
  - c. Enter the field name. See limitations below: [Available Field Properties](#).
  - d. Click **Create**.
3. Your new field is displayed in the left-hand tree and you can configure its properties in the right-hand panel.

## Available Field Properties

- **Name** - The name of the new field in the database, this must start with the **data.** prefix and can only contain letters, digits,

underscores "\_", periods ".", begin with a letter and be a minimum of 2 characters long.

Using periods "." in the Name field will create a nested structure, i.e.,

**data.newfield.newrow.alpha**

will create a field that corresponds to (assuming field type 'string'):

```
data:{
  newfield:{
    newrow:{
      alpha: ""
    }
  }
}
```

- **Type** - The type of data that you will be storing in this field, and may be:
  - **integer** - Any integer value from -2,147,483,648 to 2,147,483,647.
  - **long** - Any integer value from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
  - **float** - A floating point number.
  - **string** - Any combination of characters up to 16 KB. *To search **string** fields with the [accounts.search API](#), you can use **partial case-insensitive** search terms.*
  - **basic-string** - A string that only allows exact matching. Limited to 16K in length and allows grouping in a search query.
  - **encrypted string** - A value of type **string** that is also [encrypted](#).
  - **text** - Any combination of characters up to 64 KB. *To search **text** fields with the [accounts.search API](#), you can only use **exact match** search terms.*
  - **date** - Date values should be in the form **YYYY-MM-DD**.
  - **boolean** - Supports values of **true** / **false**.
- **Validation:** The values users are allowed to submit in this field. The validation definition changes according to the type of field and are available for **data** and **ds** fields only:
  - Numerical (integer, long, float): Specify a specific number, or a range of numbers, that users are allowed to submit. You can specify a set of values and/or ranges, separated by a comma, e.g. *3,5-10,15-20*.
  - String (string, basic-string, encrypted-string, text): Enter a regular expression pattern (regex) that must be matched in order for the field to be submitted.
  - Boolean: Use the dropdown to select whether the user is required to pass "true" or "false" in this field.
- **Write Access** - Whether this field can be updated via the client-side (i.e., using Gigya's [Web SDK](#)) or only with a signed server request.
- **Required** - Whether this field is required to contain data. All fields designated as **required** should be available on the site's *Registration* and *Registration Completion* forms, so that new users will be able to complete registration. For example, if the field **data.random** is required, and a user registers with a social network (that does not pass this field to Gigya), the Registration Completion screen will appear. Therefore, make sure to include **data.random** in the form, to allow the user to successfully submit their registration.
- **Require Double Opt-In** - For subscription fields, define whether this subscription requires [double opt-in confirmation](#), i.e., whether it requires that users confirm via email that they do, indeed, wish to subscribe to a selected subscription. .
- **Nullable** - Whether this field is allowed to accept NULL as a value.

For additional information on the above properties, see [accounts.setSchema](#) and [ds.setSchema](#).

#### Important notes:

- Once a field is saved to the schema using the **Schema Editor**, it can not be deleted using [accounts.setSchema](#). Using [account s.setSchema](#), it is only possible to delete fields that data has never been saved to (even if said data has been deleted) and it has not had a **type** defined (creating a field using the Schema Editor always assigns a **type**). If you think there is a scenario where you may not use a field and would like the ability to delete it and create another field with the same name, use [accounts.setSchema](#) to create the field without setting its **type**. Once a field is created using the Schema Editor, it can only be removed via the Schema Editor or using the [accounts.deleteSchemaFields](#) API and you can **not** create a new field using the same name.
- Once you set a field to encrypted, you cannot reverse that definition.
- Not all the above properties are available for all field types. For example, for subscription fields, you can only choose whether they are required or not.

From the Properties panel of the selected new field you can **Remove Data Field** if you do not want to save it, or **Save Changes** to the schema to make the new field permanent. You can also **Refresh** the schema or **Discard Changes** if you want to remove all unsaved changes.

Data is accurate as of: Jun 1, 2017, 12:09:34 Refresh Create Data Field

accounts

- data\*
  - globalLogins
  - metaDataRegForm003
  - newFieldInProfile01
  - newSchemaField01\*
    - newField\_two\*
  - pp\_newdata0001
  - pp\_newdata0002
  - pp\_newdata0003
  - pp\_newdata0004
  - previousLogins
  - ProfileID
  - progressive\_field\_03
  - progressive\_logins
  - progressive\_timer
  - progressive\_views
  - randomNewField3
  - sawNewScreen222
  - subscribe

Name \* data.newSchemaField01.newField\_two

Type string

Validation (regex)

Write Access clientCreate

Required

Nullable

Remove Data Field

Discard Changes Save Changes

When you are finished editing your schema, **Save Changes** to add all new fields to your site's schema.

## Deleting Fields

If no data has yet been saved to a custom data field, you can delete it.

This parameter or feature is part of our Early Adopters Program. To find out if you are eligible for participation, contact your Customer Engagement Executive by filling out a [support form](#). You can access the support page by clicking **Support** on the upper menu after logging into your [Gigya Console](#).

1. Select the data field you wish to delete, and click Delete Data Field.

The screenshot displays a data model editor interface. On the left, a tree view shows the following structure:

- accounts
  - data
    - sameNamedFieldAsGroup\_a
    - subscribe
    - terms (highlighted)
    - test\_field\_notInGroup\_01
  - profile
    - activities
    - address
    - age
    - bio
    - birthDay
    - birthMonth
    - birthYear
    - certifications
    - city
    - country
    - education
    - educationLevel
    - email

The right-hand panel shows the configuration for the selected 'terms' field:

- Name:** data.terms
- Type:** boolean
- Validation (boolean):** E. g. true
- Write Access:** clientModify
- Required
- Nullable

A **Delete Data Field** button is located at the bottom right of the properties panel.

2. Confirm the deletion.
3. The field will be deleted once you save all the changes you made in this session (when you click **Save Changes**).

## Changed Properties

When an existing field's Properties have changed, you will see an **asterisk** "\*" next to the corresponding field name in the left-hand tree, as well as all parent nodes. The process for saving or discarding the change are the same as described above. Note that you can not edit the **Name** or **Type** of an existing field.

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

[Refresh](#)

[Create Data Field](#)

The screenshot displays a configuration interface for a data field. On the left, a tree view shows the hierarchy: **accounts** > **data\*** > **newField\*** > **field\_001\***. The **field\_001\*** field is selected. The right pane shows the configuration for this field:

- Name \***: data.newField.field\_001
- Type**: string
- Validation (regex)**: (empty text box)
- Write Access**: clientCreate
- Required**:
- Nullable**:

Buttons for **Remove Data Field**, **Discard Changes**, and **Save Changes** are visible at the bottom.

[Discard Changes](#)

[Save Changes](#)

## Site Group Schema

If your site is a **member** of a **Site Group**, the only option available is to edit the **Required** property of existing fields, new fields can only be added to the site group schema via the parent's [Dashboard](#).

## Additional Information

For additional information see the following resources:

- [accounts.setSchema](#)
- [accounts.getSchema](#)
- [Accounts Profile object](#)
- [Accounts Data object](#)
- [ds.setSchema](#)
- [ds.getSchema](#)
- [ds.getTypes REST](#)
- [Subscriptions Object](#)