accounts.search REST

Description

Searches and retrieves data from Gigya's Accounts Storage using an SQL-like query. SQL queries are converted into Gigya's proprietary query language. SQL injection attacks are not possible because queries are both created by the customer and then converted by Gigya. A short delay is possible between the writing of account data and its availability in queries.

> **Note:** This method is part of the Customer Identity and the Profile Management - IDS packages. Both packages are premium platforms that require separate activation. If neither are part of your site package, please contact your Gigya Customer Engagement Executive or contact us by filling in a support form on our site. You can also access the support page by clicking "Support" on the upper menu of Gigya's site.

Example queries and responses can be viewed at accounts.search examples.

Query Syntax Specification

Gigya queries use the same syntax rules as SQL, however not all standard SQL key words are available.

- When querying for string values, value must be wrapped in double quotes. e.g., SELECT * FROM accounts WHERE name = "John Doe".
- When querying for Integer (and other non-textual fields) values, value must **not** be wrapped in quotes. e.g., SELECT * FROM accounts WHERE age = 42.
- Unsupported SQL syntax in the query string (e.g., HAVING) will produce an error.
- The query string clauses must be ordered in the following way*:
    1. Select clause
    2. From clause
    3. Where clause
    4. Filter clause
    5. Group By clause
    6. Order By clause
    7. Start clause Or/And Limit clause

    *Queries ordered differently will produce an error.
    For example:
    SELECT *, profile.age FROM accounts GROUP BY profile.age LIMIT 5 - *is a valid query*
    SELECT *, profile.age FROM accounts LIMIT 5 GROUP BY profile.age - *will produce an error*

- Encrypted fields are decrypted during searches but comparison operators (>, >=, <, <=) and regex expressions are not available on these fields. The *Contains* keyword can be used for case-insensitive searches on encrypted fields but does not support partial strings. Usernames, emails, friends' names and friends' emails are encrypted by default, additional fields may be set as encrypted by the site.
- Deleted accounts do not appear in queries.
- Query examples can be generated and query commands tested using the Identity Query Tool in Gigya's website: After signing in, go to **Reports >> User Identities >> Identity Query Tool** or click here.

select - The "select" statement accepts a comma separated list of fields or objects to retrieve. Acceptable values for this statement are:

- Field names, specifying the complete path, i.e. *data.album.photo.photoTitle_t, profile.firstName.* Specifying partial fields names (*data.album*) will return all the fields in the path.
- Object names, specifying an object type, i.e., *profile* will return all the fields in the Profile object.
- Partial field names (fields that contain only a part of the path to sub-objects, i.e., *data.album*) - will retrieve everything below that path.
- ***** - will retrieve every field in the schema.
- ***count(*)*** - if the data source is *accounts*, returns the number of accounts. If the data source is an object *type,* returns the number of objects in the data store. The result is given in the response as a single value inside the *"data"* field.
- ***as*** - create an alias (temporary title) for the returned object or field. '*SELECT profile.firstName AS contactName...'* will return a field inside the profile obj called *contactName* containing the values of profile.firstName. Example:

```
// Query:
SELECT profile.firstName AS contactName FROM accounts

// Returns
{
  "results": [
    {
      "profile": {
        "contactName": "Eric"
      }
    },
    {
      "profile": {
        "contactName": "Igor"
      }
    },
    {
      "profile": {
        "contactName": "Limor"
      }
    },
        ... snipped ...
  ],
  "objectsCount": 300,
  "totalCount": 1032,
  "statusCode": 200,
  "errorCode": 0,
  "statusReason": "OK",
  "callId": "ad24d124f11149729acdb0e7c6a6e590",
  "time": "2017-05-04T09:14:35.008Z"
}
```

- ***sum(), min(), max(), avg(), sum_of_squares(), variance(), std() -*** mathematical functions, must all be performed on the same numeric field. Fields with null values will be ignored.
  The name of the field on which the function is to be performed must be entered in the brackets. For example: '*SELECT min(profile.age) FROM accounts*'.
  - *sum* - provides a total for the field in the brackets.
  - *min/max* - minimum/maximum value for the field. If no values are found, ***min*** will return "infinity" and ***max*** will return "-infinity".
  - *avg* - average value of the field.
  - *variance* - the extent that the field's values vary.
  - *std* - standard deviation, the likelihood that values vary.

from - Name of the data source. Only one data source is supported. Account and IDS queries must state "*FROM accounts*" (accounts.search or IDS.search). Data will be retrieved from the Profile object and/or the user defined data object in the user accounts.

where - The "where" clause defines conditions for selecting items from the collection. Supported operators:

- **>** , **>=**, **<**, **<=** , **=** , ***!=*** - the left operand must be a data field name (with a proper suffix letter) and the right operand must be a constant of the same type as the field. For example: "*WHERE profile.age >= 18* ".
  - Only **=** and ***!=*** can be used with encrypted fields.
    ***Note:** The "**=**" operand is case sensitive.
- ***and*** , ***or***
- ***contains, not contains*** - may be used only on text (string) fields and arrays.
  - Text (string) fields - supports standard full text search capabilities. **Contains** is case sensitive, except when used on encrypted fields. The left operand must be a text field name and the right operand must be a constant string. You can search for a specific word within the string, for example: '*WHERE data.about_t CONTAINS "music"* '. Underscores are treated as separators between words.

If you want to perform a search on an encrypted field, you must enter the full string. Encrypted fields include the fields you encrypted and also fields that Gigya encrypts.

- Arrays - the left operand must be an array field name and the right operand must be a constant of the same type as the array values. The array field name must have a suffix denoting the type of the arrays values. For example: '*WHERE data.hobbies_s CONTAINS "swimming"* '.

Note: You can only search words that are part of a sentence, you cannot search for parts of a word.

- ***in()*** - only retrieve items if the field contains one of the list values. For example: *'SELECT * FROM accounts WHERE profile.firstName IN ("Frank", "Dean", "Sammy")*' will return users with the specified first names.
- ***is null*** , ***is not null***
- ***not***
- ***regex*** ('<regex-pattern>') - defines a search term using regex formatting. The regex syntax can be found in: Regular Expression Language Reference. Regex patterns cannot be used on encrypted fields.

order by - The "*order by*" clause specifies a list of fields by which to sort the result objects. You can not use ORDER BY on encrypted fields.

limit - Using the "limit" clause, you may specify the maximum number of returned result objects. If not specified, the default is 300. The maximum limit value accepted is **10000**. If the search is sent with openCursor = true, *limit* will set the batch size. Please note, when using a cursor, the number of results in a batch is not guaranteed.

start - The "start" clause (not an SQL standard clause) may be used for paging. The clause specifies the start index from which to return result objects. The maximum start value accepted is **5000**.

The '*select - from - where - order by*' query creates an (internal) indexed list of objects. By using the *start* and *limit* clauses, you will receive a subset of this list, starting with the *start* index and ending with *start+limit* index.

Notes:

- When implementing paging, there is no guarantee against duplications or that recently added data will show up in the query results.
- **start** cannot be used with **openCursor**.

Counters - Counter data is only available if the data source statement ("**from**") specifies it, i.e., "*FROM accounts WITH counters*". For example: *' SELECT * FROM accounts WITH counters LIMIT 5'*. To view only the counter fields, *'SELECT counters FROM accounts WITH counters'*. To specify a particular field in the counters object, use "*counters.class*" or "*counters.path*" etc. Note that counter data is not retrievable using the accounts.search API until 1-24 hours after it is written. To get real-time counter data, use accounts.getCounters.

Two SQL keywords exist for use with counter fields (they are not available with other fields): **ifElement** is used for setting multiple conditions on a single counter's different fields and **filter counters** is used to restrict the counters returned with an account object.

- ***filter counters by class="****<class name>"* - restricts the counter types returned with the account object to those listed in the filter statement. For example: *filter counters by class="shares"*, returns only the shares counters.
- ***ifElement(****counters****,*** *<if statement relating to different elements of a single counter>****)*** - this function allows you to apply conditions to several elements within the same counter, for example: *ifElement(counters, class="purchases" and period = "total" and value > 100).* Wh
on true the ifElement expression returns 1.

Unable to render {include} The included page could not be found.

Query Optimization

Below are a few points to note regarding query optimization:

1. Query execution is based on clause position and is executed from left to right.
2. Place clauses that have the greatest impact on records returned at the beginning of your SQL statement. For example, to retrieve a list of male users over the age of 25:

   This is because filtering first by gender automatically reduces the result set by half, so the server only needs to run the next filter on half of the overall population.
3. A NOT clause (NOT or !) is executed on a single statement immediately to it's right, after analyzing the statement. A single statement can hold several conditions inside parentheses.
4. Date ranges are calculated much more efficiently using a *timestamp* field rather than a *date* field.
5. Use of regex is computationally intensive and can significantly increase response time.
6. AND clauses take precedence over OR clauses (i.e., AND clauses are executed before OR clauses).
7. Use parentheses to modify default precedence (e.g., to execute an OR operation before an AND operation).

Pagination

When running long queries (>5,000 records returned), it's best practice to paginate your results using cursors. If you do not use cursors, **results are limited to a total of 5,000 records per query** (not just per page).

To use cursors, during the first request, pass *query=<query to execute>* and *openCursor=true*. The response will include the *nextCursorId* field, containing a cursor ID to be used in the next request. On subsequent requests, pass *cursorId=<last response's nextCursorId>* and **do not** submit

the query again. The absence of the *nextCursorId* field in a response indicates the end of the result set.

> When using openCursor, you cannot use 'START'.

Request URL

Where **<Data_Center>** is:

- **`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.

If you are not sure of your site's data center, see Finding Your Data Center.

Note: Use a POST request rather than GET if you are using a direct REST call.

Parameters

Required	Name	Type	Description
	query	string	An SQL-like query specifying the data to retrieve. Please refer to the Query language specification section above.
The following parameters are **Required** *only* when calling the search method from **client** side (i.e., Mobile SDKs):			
	querySig	string	An HMAC_SHA1 signature proving that the search call is in fact coming from your client application, in order to prevent fraud. Follow the instructions in Constructing a Signature using the following base-string: ***query*** + "_" + ***expTime*** . When using cursors, this parameter should only be sent with the initial request and omitted from subsequent requests. ***When using querySig the openCursor property is not supported. openCursor is only supported in server-to-server calls that include a userKey and secret.**
	expTime	string	The GMT time when the signature, provided in the ***UIDSig*** parameter, should expire. The expected format is the Unix time format including milliseconds (i.e., the number of seconds since Jan. 1st 1970 * 1000). Gigya checks the time when the search request is received. If the time succeeds expTime, the request is considered forged.
	openCursor	Boolean	When set to true, the search response will include, in addition to the first page, another field named **nextCursorId**, which is used to fetch the next batch of results. This parameter should only be used on the first request and later should be removed from the request. When openCursor is active, the *Limit* clause sets the number of results returned in the batch and should not be larger than 1000 (one thousand). **Notes:** • When using a cursor with a *Limit* set, the number of results in a batch is not guaranteed. • You cannot use a cursor if you have a *group by* or when using *'start'*. • openCursor is not supported when using querySig and can only be used in server-to-server calls that include a userKey and secret.
	cursorId	string	The cursor ID that contains the **nextCursorId** value received in the first search call. **Notes:** • You cannot pass both **cursorId** and **query** on the same request - **cursorId** brings the next page for the search for which it was opened. Also, the time between search requests using a **cursorId** must not exceed 5 minutes (300 seconds). • Each request should contain a different **cursorId** obtained from the response of the **previous** request (not the first) using the **nextCursorId** field. The exception to this rule is when a request fails or when a particular result set needs to be resent; in this case, resend the same **cursorID** (as long as it has not expired) to receive its associated result set.
	timeout	integer	The timeout for the request (in milliseconds). Default value is 20000 (20 seconds). Maximum allowed value is 60000 (60 seconds).
	restrictedQuery	string	An SQL-like query specifying the data to retrieve. When using this parameter, the query specified must meet the regex criteria defined for the user making this call.
	accountTypes	string	The type of account to retrieve: full or lite. Acceptable values: • **full** (the default value) • **lite** • **full,lite**

format	string	Determines the format of the response. The options are: • *json* (default) • *jsonp* - if the format is jsonp then you are required to define a *callback* method (see parameter below).
callback	string	This parameter is *required* only when the *format* parameter is set to *jsonp* (see above). In such cases this parameter should define the name of the callback method to be called in the response, along with the jsonp response data.
context	string/JSON	This parameter may be used to pass data through the current method and return it, unchanged, within the response.
dontHandleScreenSet	Boolean	This parameter may be used in order to suppress the showing of screen-sets as a result of API calls. Default is **false**.
httpStatusCodes	Boolean	The default value of this parameter is *false*, which means that the HTTP status code in Gigya's response is always 200 (OK), even if an error occurs. The error code and message is given within the response data (see below). If this parameter is set to *true*, the HTTP status code in Gigya's response would reflect an error, if one occurred.

Authorization Parameters

Each REST API request must contain identification and authorization parameters.

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 IP 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. A non-exhaustive list of REST APIs that this may apply to are as follows:

- accounts.login
- socialize.login
- accounts.notifyLogin
- socialize.notifyLogin
- accounts.finalizeRegistration
- accounts.linkAccounts

Please refer to the Authorization Parameters section for details.

Sample Requests

Search from accounts

Response Data

Field	Type	Description

errorCode	integer	The result code of the operation. Code '0' indicates success, any other number indicates failure. For a complete list of error codes, see the Error Codes table.
errorMessage	string	A short textual description of an error, associated with the errorCode, for logging purposes. This field will appear in the response only in case of an error.
errorDetails	string	This field will appear in the response only in case of an error and will contain the exception info, if available.
fullEventName	string	The full name of the event that triggered the response. This is an internally used parameter that is not always returned and **should not** be relied upon by your implementation.
callId	string	Unique identifier of the transaction, for debugging purposes.
time	string	The time of the response represented in ISO 8601 format, i.e., yyyy-mm-dd-Thh:MM:ss.SSSZ or
statusCode	integer	The HTTP response code of the operation. Code '200' indicates success. This property is deprecated and only returned for backward compatibility.
statusReason	string	A brief explanation of the status code. This property is deprecated and only returned for backward compatibility.

results	Array	An array of Account objects (full or partial objects, based on your selected clause), retrieved from Gigya's Accounts Storage.
nextCursorId	string	Used to fetch the next batch of results. This parameter is not returned on the last batch of results, its absence means that the result set is finished.
objectsCount	integer	The number of objects returned in the *"results"* array.
totalCount	integer	The total number of objects that satisfy the query in the DB. This is useful when fetching a limited amount using the "*limit*" parameter.

Account Object

Field	Type	Description
UID	string	The unique user ID. This user ID should be used for login verification. See User.UID for more information.
UIDSignature	string	The signature that should be used for login verification. See User.UID for more information.
signatureTimestamp	string	The GMT time of the response in UNIX time format, i.e., the number of seconds since Jan. 1st 1970. The timestamp should be used for login verification. See User.UID for more information.
created	string	The UTC time the account was created in ISO 8601 format, e.g., "1997-07-16T19:20:30Z".
createdTimestamp	integer	The UTC time the account was created in Unix time format including milliseconds (i.e., the number of seconds since Jan. 1st 1970 * 1000).
data	JSON object	Custom data. Any data that you want to store regarding the user that isn't part of the Profile object.
emails	JSON object	The email addresses belonging to the user. This includes the following fields: • ***verified*** - an array of strings representing the user's verified email addresses • ***unverified*** - an array of strings representing the user's unverified email addresses. Note: ***emails*** must be specified explicitly in the ***include*** parameter in order to be included in the response.

identities	array	An array of Identity Objects, each object represents a user's social identity. Each Identity Object contains imported data from a social network that the user has connected to. **Note:** You must explicitly specify ***identities*** within the ***include*** parameter for them to be included in the response: ***identities-active*** , ***identities-all***, or ***identities-global*** to return only active identities, all identities of a site, or all identities of a site group, respectively. Be advised that if a user registers to your site using a **Social Identity**, then goes through the **Forgot Password** flow, a **Site Login** is added to their account, however, a **Site Identity** is *not*. A **Site Identity** can only be created when accounts.setAccountInfo is called on the user's account.
iRank	*integer*	*Influencer rank of the user. This property is deprecated and will always return 0.*
isActive	Boolean	Indicates whether the account is active. The account is active once the user creates it even without finalizing it. The account can be deactivated, but it will still be registered if the registration process has been finalized. If isActive==false the user cannot log in, however any currently active sessions remain valid.
isLockedOut	Boolean	Indicates whether the account is currently locked out. This parameter is not included in the response by default, and is not returned at all from accounts.search. If you wish to include it in a response, specify it as a value of the **include** parameter.
isRegistered	Boolean	Indicates whether the user is registered. The user is registered once his registration has been finalized.
isVerified	Boolean	Indicates whether the account email is verified.
lastLogin	string	The time of the last login of the user in ISO 8601 format, e.g., "1997-07-16T19:20:30Z".

lastLoginLocation	JSON object	The user's last login location. This includes the following fields:
		• **country** - a string representing the two-character country code.
		• **state** - a string representing the state, where available.
		• **city** - a string representing the city name.
		• **coordinates** - an object containing:
		  • **lat** - a double representing the latitude of the center of the city.
		  • **lon** - a double representing the longitude of the center of the city.

lastLoginTimestamp	integer	The UTC time of the last login of the user in Unix time format including milliseconds (i.e., the number of seconds since Jan. 1st 1970 * 1000).
lastUpdated	string	The UTC time when user profile, preferences, or subscriptions data was last updated (either full or partial update) in ISO 8601 format, e.g., "2017-07-16T19:20:30Z".
lastUpdatedTimestamp	integer	The UTC time when the last update of the object occurred (either full or partial update) in Unix time including milliseconds, based on when the 'lastUpdated', 'Report AccountsFirstLogin' or 'AccountsReturnedLogin' events are fired.
loginIDs	JSON object	The user's login identifiers. This includes the following fields: • ***username*** - a string representing the username • ***emails*** - an array of strings representing email addresses • ***unverifiedEmails*** - an array of strings representing email addresses that were not validated Note: ***loginIDs*** must be specified explicitly in the ***include*** parameter in order to be included in the response.
loginProvider	string	The name of the provider that the user used in order to login.
oldestDataUpdated	string	The UTC time when the oldest data of the object was refreshed in ISO 8601 format, e.g., "1997-07-16T19:20:30Z".
oldestDataUpdatedTimestamp	integer	The UTC time when the oldest data of the object was refreshed in Unix time format including milliseconds (i.e., the number of seconds since Jan. 1st 1970 * 1000).

password	JSON object	The user's **Site** account password details. Includes the following:
		• ***hash*** - the hashed password
		• ***hashSettings*** - object includes:
		  • **algorithm** - Represents the hash algorithm used to encrypt the password.
		  • **rounds** - Represents the number of iterations to perform the hashing.
		  • **salt** - Represents the BASE64 encoded value of the salt.
		  • **format** - Represents the template for merging clear-text passwords. This is only returned if the **pwHashFormat** parameter was set during account import and until the user's first login to Gigya (when the user's password is rehashed per the site's settings). See the RaaS Import Guide for additional information.

UIDSignature	*string*	***This property is deprecated in server to server REST calls!*** *The signature that should be used for login verification. See User.UID for more information.*
signatureTimestamp	*string*	***This property is deprecated in server to server REST calls!*** *The GMT time of the response in UNIX time format, i.e., the number of seconds since Jan. 1st 1970. The timestamp should be used for login verification. See User.UID for more information.*
phoneNumber	string	The Phone Number ID, if the account uses Phone Number Login.
preferences	Preferences object	The user's preferences information as described in the Preferences Object. To have this data returned in the response it must be specifically requested using the **include** parameter .
profile	Profile object	The user's profile information as described in the object. The ***profile*** is returned in the response by default, but if the ***include*** parameter is used to specify other fields that should be provided in the response, the ***profile*** must also be specified explicitly in the ***include*** par ameter.
rbaPolicy	JSON object	The current RBA Policy defined for the specified user. Properties include: • **riskPolicy** - Determines the rule set from the defined rulesSets configured in accounts.rba.setPolicy or one of the default policies. • **riskPolicyLocked** - Determines whether the user can change their own riskPolicy. If true, only an admin can change the user's riskPolicy.
registered	string	The UTC time when the ***isRegistered*** parameter was set to true in ISO 8601 format, e.g., "1997-07-16T19:20:30Z".
registeredTimestamp	string	The GMT time when the ***isRegistered*** parameter was set to true in UNIX time format, including milliseconds.
regSource	string	A string representing the source of the registration. Can be used to set varying destination pages in accounts.setPolicies.
socialProviders	string	A comma-separated list of the names of the providers to which the user is connected/logged in.
subscriptions	Subscriptions Object	The user's subscription information.
userInfo	*User object*	*The Gigya User object. This property is deprecated and should not be relied upon.*
verified	string	The UTC time when the ***isVerified*** parameter was set to true in ISO 8601 format, e.g., "1997-07-16T19:20:30Z".
verifiedTimestamp	string	The GMT time when the ***isVerified*** parameter was set to true in Unix time format including milliseconds (i.e., the number of seconds since Jan. 1st 1970 * 1000).

A field that does not contain data will not appear in the response.

Response Example

```
{
      "statusCode": 200,
```

```
"errorCode": 0,
"statusReason": "OK",
"callId": "31ba039fb8d340ceb2f43d52c89bf187",
"time": "2015-03-22T11:42:25.943Z",
"results": [{
      "UID": "17490",
      "isRegistered": true,
      "registeredTimestamp": 1344525120445,
      "registered": "2012-08-09T15:12:00.445Z",
      "isActive": true,
      "isVerified": false,
      "iRank": 0,
      "loginIDs": {
          "username": "h17490@gmail.com",
          "emails": [],
          "unverifiedEmails": []
      },
      "emails": {
          "verified": [],
          "unverified": ["h17490@gmail.com"]
      },
      "socialProviders": "site",
      "profile": {
          "email": "rastropovich17490@gmail.com",
          "firstName": "Joe",
          "lastName": "Smith",
          "age" : "31",
          "gender" : "m",
          "country" : "US"
      },
      "identities": [{
          "provider": "site",
          "providerUID": "17490",
          "isLoginIdentity": false,
          "gender": "",
          "email": "h17490@gmail.com",
          "allowsLogin": false,
          "isExpiredSession": false,
          "lastUpdated": "2012-08-09T15:12:00.302Z",
          "lastUpdatedTimestamp": 1344525120302,
          "oldestDataUpdated": "2012-08-09T15:12:00.302Z",
          "oldestDataUpdatedTimestamp": 1344525120302}],
      "data": {},
      "created": "2012-08-09T15:12:00.297Z",
      "createdTimestamp": 1344525120297,
      "lastLogin": "0001-01-01T00:00:00Z",
      "lastLoginTimestamp": 0,
      "lastUpdated": "2012-08-09T15:12:00.302Z",
      "lastUpdatedTimestamp": 1344525120302,
      "oldestDataUpdated": "2012-08-09T15:12:00.302Z",
      "oldestDataUpdatedTimestamp": 1344525120302},
  {
      "UID": "10067",
```

```
        "isRegistered": true,
        "isActive": true,
        "isVerified": false,
        "iRank": 0,
        "loginIDs": {
            "username": "vich@gmail.com",
            "emails": [],
            "unverifiedEmails": []
        },
        "emails": {
            "verified": [],
            "unverified": ["vich@gmail.com"]
        },
        "socialProviders": "site",
        "profile": {
            "email": "vich10067@gmail.com",
            "firstName": "David",
            "lastName": "Cohen",
            "age" : "50",
            "gender" : "m",
            "country" : "Canada"
        },
        "identities": [{
            "provider": "site",
            "providerUID": "10067",
            "isLoginIdentity": false,
            "gender": "",
            "email": "vich@gmail.com",
            "allowsLogin": false,
            "isExpiredSession": false,
            "lastUpdated": "2012-08-09T15:02:56.969Z",
            "lastUpdatedTimestamp": 1344524576969,
            "oldestDataUpdated": "2012-08-09T15:02:56.969Z",
            "oldestDataUpdatedTimestamp": 1344524576969}],
        "data": {},
        "password": {
            "hash": "YG8PL6PwxlH0+KbUb4vG3w==",
            "hashSettings": {
              "algorithm": "pbkdf2",
              "rounds": 5000,
              "salt": "iIj9T09VwfcvLv/0D7rFkA=="
            }
          },
        "tfaStatus": "forceOff",
        "created": "2012-08-09T15:02:56.961Z",
        "createdTimestamp": 1344524576961,
        "lastLogin": "0001-01-01T00:00:00Z",
        "lastLoginTimestamp": 0,
        "lastUpdated": "2012-08-09T15:02:56.969Z",
        "lastUpdatedTimestamp": 1344524576969,
        "oldestDataUpdated": "2012-08-09T15:02:56.969Z",
        "oldestDataUpdatedTimestamp": 1344524576969
    },
```

```
            ...
        ]
}
```