Rosetta Stone API Services

Updated 06/09/2022

Table of Contents

Preface

The Catalyst API service provides a programming interface to report user progress, create, update, and modify users, and assign/unassign licensing.
The Fluency Builder API service provides course assignment, unassignment, and deletion. Both API services are based on the GraphQL query language.

This document will provide the following information:

- Generating and refreshing an access token for authorization
- Sample queries to call org, user, license, and progress information
- Sample mutations to add, modify, delete, un/license users
- Sample mutations to assign, unassign, and delete Fluency Builder courses
- Additional reference material from external resources

Accessing API Services

Catalyst and Fluency Builder APIs act independently by endpoint but are used for the same Catalyst organization:

- Catalyst API: *https://rsm2-api.rosettastone.com/graphql*
- Fluency Builder: *https://gaia-server.rosettastone.com/graphql*

Accessing either API service will require:

- A Catalyst organization administrator account
- Access token generated using the administrator account

All Catalyst and Fluency Builder API calls require a Catalyst/tully Oauth2 token. The same token is interchangeable between Catalyst and Fluency Builder API..

Property Name	**value**
AccessTokenUri	https://tully.rosettastone.com/oauth/token
grant_type	password
client_id	client.rsm2api
username	provided by Rosetta Stone must have Administrator Tools license, i.e., Organization Admin
password	password

Generating an access token

1. Send an HTTP request to the AccessTokenUri using the administrator username and password. Examples in URL and cURL:

   URL:

   ```
   https://tully.rosettastone.com/oauth/token?grant_type=password&client_id=client.rsm2api&scop
   e=&username=JohnMcClain%40nakatomiplaza.com&password=Yippee-ki-ay
   ```

   cURL:

   ```
   curl --request POST \
     --url 'https://tully.rosettastone.com/oauth/token' \
     --header 'content-type: application/x-www-form-urlencoded' \
     --data grant_type=password \
     --data 'client_id=client.rsm2api' \
     --data username=JohnMcClain%40nakatomiplaza.com
     --data password=Yippee-ki-ay
   ```

2. The access token will be the first section of the response

   ```
   {"access_token":"e0daa57e-47d7-478f-8cb5-6a472fef8f2d","token_type":"bearer"...
   ```

A token will remain valid for 8 hours from the initial authorization. After a token expires, requests to the rsm2-api graphql endpoint will return HTTP 401 and an additional token may be requested.

Refreshing an access token

If you know your token is going to expire soon (see "expires_in" property of the token response), you may refresh it using grant_type "refresh_token"

This request is very similar to oauth2 password grant. Change the grant_type to "password", and use the refresh_token from the previous response instead of the username and password.

URL:

```
https://tully.rosettastone.com/oauth/token?grant_type=refresh_token&client_id=client.rsm2api&
refresh_token=e00af2a7-95db-4ccb-8563-0c6578ebf0b9
```

cURL

```
curl --request POST \
  --url 'https://tully.rosettastone.com/oauth/token' \
  --header 'content-type: application/x-www-form-urlencoded' \
  --data grant_type=refresh_token \
  --data 'client_id=client.rsm2api' \
  --data refresh_token=e00af2a7-95db-4ccb-8563-0c6578ebf0b9
```

Catalyst API

Using API methods

The Catalyst API utilizes GraphQL. While typical REST APIs require loading from multiple URLs, GraphQL APIs get all the data needed in a single request. The Catalyst API's methods include:

- Query
  - Getting user information
  - Getting organization information
  - Getting license information
- Mutation
  - Creating users
  - Registering users
  - Updating users
  - Deleting users
  - Manage license assignment

A query's response is formed by the object and fields submitted within the request. The examples that follow provide typical objects and fields within the submission, however, use-case will vary and more or less information may need to be retrieved. For a full list of objects and fields, refer to the GraphQL Catalyst API schema.

A word about licensing and product assignment

LicensePools shows various details on licenses available or assigned to users. This detailed information does not show which product is in use but which license is assigned. To determine if the user has been assigned Fluency Builder or Foundations, refer to the "level" attribute of "languagePrograms". Levels A1-A2 are assigned Foundations while B1 and above are assigned. Refer to the schema for additional details.

Example queries

myOrg
Description:
This request will return the organization information id, name, namespace, when it was created. Additionally, licensing information was also called which shows licensing information within the listed fields.
	query {
	  myOrg {
	  id
	  name
	  namespace
	  createdAt
	  licensePools {
	  id
	  productId
	  productName
	  salesPackageId
	  salesPackageName
	  poNumber
	  createdAt
	  totalLicenseCount
	  assignedLicenseCount
	  licenseModel
	  termStart
	  termEnd
	  }
	  }
	}

getUser

Description:

Request a single user’s information. Data items here include usageGoal and languagePrograms with all-time, 30 day, month-to-date, and custom range usage summary.

Usage will aggregate for the language of the program.
Only the current language program is returned.

languagePrograms:

allTimeUsageSummary - Aggregate of usage for all time
last30daysUsageSummary - Aggregate of usage for previous calendar 30 days from current date.
monthToDateUsageSummary - Aggregate of usage from the 1st of the month to current date.
timePeriodUsageSummary - Time period usage summary from last 'timePeriodDays' days. If the variable is omitted, it defaults to 0, resulting in usage for the current date only.

```
query {
  user(id: "cc646202-23a4-4d31-af7e-67202753f4db") {
    id
    email
    status
    firstName
    lastName
    usageGoalDays
    usageGoalMinutes
    languagePrograms (timePeriodDays: 7){
      languageOfStudy
      level
      lastUsageAt
      allTimeUsageSummary {
        totalMinutes
        desktopLessonMinutes
        mobileLessonMinutes
        tutoringMinutes
      }
      last30daysUsageSummary {
        totalMinutes
        desktopLessonMinutes
        mobileLessonMinutes
        tutoringMinutes
      }
      monthToDateUsageSummary {
        totalMinutes
        desktopLessonMinutes
        mobileLessonMinutes
        tutoringMinutes
      }
      timePeriodUsageSummary {
        totalMinutes
        desktopLessonMinutes
        mobileLessonMinutes
        tutoringMinutes
      }
    }
    licenses {
      assignedAt
      licensePool {
        productName
        termStart
        termEnd
      }
    }
  }
}
```

userBySsoId

Search users by SSOID to return user details.

```
query userBySsoId($ssoId: String!) {
    userBySsoId(ssoId: $ssoId) {
        id
        email
        createdAt
        updatedAt
        firstName
        lastName
        groupIds
        interfaceLanguage
        voiceType
        timezone
        originLanguage
        registrationDate
        status
        usageGoalDays
        usageGoalMinutes
        userField1
        userField2
        userField3
        userField4
        userField5
        userField6
        notes
        ssoId
    }
}
```

Variables:

```
{"ssoId": "SSO_ID_1"}
```

MyOrgUsers

Request data on all users within the organization.

Note: This query requires use of pagination for orgs with more than 100 users

```
query MyOrgUsers ($first: Int, $after: String) {
  myOrg {
    users(first: $first after: $after) {
      nodes{ id email firstName lastName}
      pageInfo {
        endCursor
        hasNextPage
      }
      totalCount
    }
  }
}
```

Variables:

```
{"first": 100, "after": null}
```

userSearch

This example only searches by email, but may also search by first or last names.

GroupId may be your org Id.

Results are additive (logical OR) meaning including email, firstName, and lastName will return users matching any one of those values.

```
query userSearchQuery($groupId: ID!, $email: String!) {
  userSearch(
    groupId: $groupId
    email: $email
  ) { id email}
}
```

Variables:

```
{"groupId": "1fe9b685-4451-48d7-b3f2-f79bd5553638", "email": "HansGruber@shellco.com"}
```

GroupUsers
Find all users within a group by groupId.

Note: this query requires use of pagination for orgs with more than 100 users.

```
query GroupUsers ($groupId: ID!, $first: Int, $after: String) {
  myOrg {
    groups (groupId: $groupId) {
      id
      name
      users(first: $first after: $after) {
        nodes{ id email firstName lastName}
        pageInfo {
          endCursor
          hasNextPage
        }
        totalCount
      }
    }
  }
}
```

Variables:
```
{"groupId": "11398efc-b9c0-41e7-8167-e5e4857324f2", "first": 100, "after": null}
```

MyOrgTestResults
Placements, self assessment, and proficiency test results can be retrieved off the user object

```
query MyOrgTestResults($first: Int, $after: String) {
  myOrg {
    users(first: $first, after: $after) {
      nodes {
        id
        email
        testResults {
          testName
          language
          cefr
          scaledScore
          completedAt
          timeSpentSecs
        }
      }
      pageInfo {
        endCursor
        hasNextPage
      }
      totalCount
    }
  }
}
```

Variables:
```
{
  "input" : {
    "userId" : "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80",
    "languageOfStudy": "de_DE",
    "clientMutationId": "foobar"
  }
}
```

fragment
Fragments can be defined and used so that you can get the same UserFields (etc), in multiple use cases Org.Users, user by id, searchUsers, etc
	fragment UserProfile on User {
	  id
	  email
	  createdAt
	  updatedAt
	  firstName
	  lastName
	  groupIds
	  interfaceLanguage
	  voiceType
	  timezone
	  originLanguage
	  registrationDate
	  status
	  usageGoalDays
	  usageGoalMinutes
	  userField1
	  userField2
	  userField3
	  userField4
	  userField5
	  userField6
	  notes
	}
	
	query {
	  myOrg {
	  id
	  name
	  users {
	  nodes {
	  ...UserProfile
	  licenses {licensePool {productName termStart termEnd}}
	  }
	  }
	  }
	}

Example mutations

CreateUser

Note: the user.id returned by this mutation is generated by the system, and will be needed to perform any additional operations on this user. Language codes are available in the Catalyst API GraphQL schema.

```
mutation CreateUserMutation($input: CreateUserInput!) {
  createUser(input: $input) {
    user {
      id
      email
      status
      languagePrograms {
        languageOfStudy
      }
    }
  }
}
```

Variables:

```
{
    "input": {
        "email": "HansGruber@shellco.com",
        "firstName": "test",
        "interfaceLanguage": "en_US",
        "languageOfStudy": "LET_LEARNER_SELECT",
        "lastName": "test8",
        "notes": "created via GraphQL mutation via rsm2-api",
        "orgId": "4fe9b685-4451-48d7-b3f2-f79bd5553638",
        "userField1": "LET_LEARNER_SELECT"
        "ssoId": "123456"
    }
}
```

updateUser

Update an existing user by userId.

```
mutation updateUser($input: UpdateUserInput!) {
  updateUser(input: $input) {
    user {
      id
      email
      firstName
      lastName
      interfaceLanguage
      status
      userField1
      userField2
      userField3
      userField4
      userField5
      userField6
      notes
      ssoId
      updatedAt
    }
  }
}
```

Variables: (include only properties you wish to change)

```
{
  "input": {
    "userId": "6754bed3-e2c8-43c9-afde-91812a27cf7f",
    "email": "HansGruber@shellco.com",
    "firstName": "Hans",
    "lastName": "Gruber",
    "interfaceLanguage": "es_419",
    "userField1": "Homeroom 878",
    "userField2": "Emp112233",
    "notes": "Duly noted",
    "ssoId": "123456"
  }
}
```

deleteUser
Delete a user by userId

Advisory:
User accounts are soft-deleted. Use ***createUser*** to undelete the user with the same email address. The user profile will default as if new and unregistered until updates are applied either through API or user registration. This applies to notes, custom userfields, and SSOID. However, product usage data remains intact.

```
mutation deleteUserMutation($input: DeleteUserInput!) {
  deleteUser(input: $input) {
    org {
      id
      name
      namespace
    }
  }
}
```

Variables:
```
{"input": {"userId": "f4c7d52b-0e82-4e77-b54b-ad149a2a5ef0"}}
```

assignLicense
Assign a license to a user by userId and licensePoolId

Advisory:
Only registered users may be assigned a license.

```
mutation assignLicenseMutation($input: AssignLicenseInput!) {
  assignLicense(input: $input) {
    user {
      id
      licenses {
        assignedAt
        licensePool {
          id
        }
      }
    }
  }
}
```

Variables:
```
{"input": {"userId": "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80", "licensePoolId": "SalesPackageConfig.ba621168-3cac-43ef-ac34-cb7709a72d1b"}}
```

unassignLicense
Unassign a license from a user by userId and licensePoolId

```
mutation unassignLicenseMutation($input: UnassignLicenseInput!) {
  unassignLicense(input: $input) {
    user {
      id
      licenses {
        assignedAt
        licensePool {
          id
        }
      }
    }
  }
}
```

Variables:
```
{"input": {"userId": "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80", "licensePoolId": "SalesPackageConfig.ba621168-3cac-43ef-ac34-cb7709a72d1b"}}
```

sendEmailNotificationRegistration
Send the registration email to a user by userId.
See ***registerUser*** to register the user without user involvement.

```
Mutation sendEmailNotificationRegistrationMutation($input:
SendEmailNotificationRegistrationInput!) {
  sendEmailNotificationRegistration(input: $input) {
    user {email}
  }
}
```

Variables:
```
{"input": {"userId": "0f4f8b3d-8cb9-423b-8aaf-547bd7ee9de1"}}
```

registerUser

Register a user, skipping the registration email and user's involvement in registration. User registration is required to enable the account for use and license assignment. This is an alternate path to sendEmailNotificationRegistration mutation.

```
mutation registerUserMutation($input: RegisterUserInput!){
  registerUser(input: $input) {
    user {
      id
      status
      timezone
    }
  }
}
```

Variables:

```
{
  "input": {
    "userId": "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80",
    "password": "foobar",
    "voiceType": "MALE",
    "originLanguage": "en-US",
    "timezone": "America/New_York"
  }
}
```

removeUserFromGroup

Unassign a user from a group by userId and groupId.

```
mutation removeUserFromGroupMutation($input: RemoveUserFromGroupInput!)
{
  removeUserFromGroup(input: $input) {
    user {
      id
      groupIds
    }
  }
}
```

Variables:

```
{"input": {"userId": "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80", "groupId": "31398efc-b9c0-41e7-8167-e5e4857324f2"}}
```

addUserToGroup

Assign a user to a group by userId and groupId

```
mutation addUserToGroupMutation($input: AddUserToGroupInput!) {
  addUserToGroup(input: $input) {
    user {id groupIds}
  }
}
```

Variables:

```
{"input" : {"userId" : "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80", "groupId": "31398efc-b9c0-41e7-8167-e5e4857324f2"}}
```

overrideLanguageProgramLevel

Set or update an existing language level (CEFR) for any user by userId.

```
mutation overrideLanguageProgramLevelMutation($input:
OverrideLanguageProgramLevelInput!) {
  overrideLanguageProgramLevel(input: $input) {
    user {
      id
      email
      languagePrograms {
        id
        languageOfStudy
        level
      }
    }
  }
}
```

Variables:

```
{"input": {"userId": "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80", "level": "B1"}}
```

newLanguageProgram

Set the language of study for any user by userId.

Language codes are available in the Catalyst API GraphQL schema.

```
mutation NewLanguageProgramMutation($input: NewLanguageProgramInput!) {
  newLanguageProgram(input: $input) {
    clientMutationId
    user {
      id
      email
      languagePrograms {
        id
        languageOfStudy
        level
      }
    }
  }
}
```

Variables:

```
{
  "input" : {
    "userId" : "2b1aed7a-559b-4f31-8d2f-9ec1cfdc3a80",
    "languageOfStudy": "de_DE",
    "clientMutationId": "foobar"
  }
}
```

createGroup

Create a new group.

```
mutation CreateGroupMutation($input: CreateGroupInput!) {
  createGroup(input: $input) {
    group {
      id
      name
      description
    }
  }
}
```

Variables:

```
{
  "input": {
    "groupName": "test api group",
    "groupDescription": "test group description"
  }
}
```

deleteGroup

Delete an existing group.

```
mutation DeleteGroupMutation($input: DeleteGroupInput!) {
  deleteGroup(input: $input) {
    wasDeleted
  }
}
```

Variables:

```
{
  "input": {
    "groupId": "f60008f9-d2ad-40c7-9b58-4ed7a624f149"
  }
}
```

Use Cases

Obtaining UserID

In order to modify a user's attributes, language program, or licensing, a "userId" is required as a variable. There are several ways to obtain the "userId":

- Locally store return from myOrgUsers
- Locally store return from createUser at user-creation
- Locate userId using userSearch if email address is known
- Locate userId using SSOID

Any of these methods may be used in conjunction with any other, however, certain methods are better suited depending on what needs to be achieved.

Store attributes of every user for later use in queries requiring userId

Perform MyOrgUsers with desired attributes to return a full roster. The results can then be stored locally for reference by any calls requiring userId. This method works well when the organization has existing users prior to API implementation.

Call	Response

```
query MyOrgUsers ($first: Int, $after: String) {
  myOrg {
    users(first: $first after: $after) {
      nodes{ id email firstName lastName}
      pageInfo {
        endCursor
        hasNextPage
      }
      totalCount
    }
  }
}
```

```
{
  "data": {
    "myOrg": {
      "users": {
        "nodes": [
          {
            "id": "f30f630f-91e1-48a3-af40-c1c55e8f58dc",
            "email": "HansGruber@shellco.com",
            "firstName": "Catalyst",
            "lastName": "Admin"
          },
          {
            "id": "cc14ede0-dc71-4e68-b845-13597525858a",
            "email": "McClain@shellco.com",
            "firstName": "Catalyst",
            "lastName": "Admin Group"
          },
          .....
```

Append new-user attributes to existing locally stored inventory

Perform createUser with “id” as a returning value. The results can be appended to the locally stored roster for reference by any call requiring userId.

Call

```
mutation CreateUserMutation($input: CreateUserInput!) {
  createUser(input: $input) {
    user {
      id
      email
      status
    }
  }
}
```

Response

```
{
  "data": {
    "createUser": {
      "user": {
        "id": "d5339a8a-6bab-4b57-bb27-7c163e78967f"
        "email: "HansGruber@shellco.com"
        "status": "UNREGISTERED"
      }
    }
  }
}
```

Variables:

```
{
    "input": {
        "email": "HansGruber@shellco.com",
        "firstName": "test",
        "interfaceLanguage": "en_US",
        "languageOfStudy": "LET_LEARNER_SELECT",
        "lastName": "test8",
        "notes": "created via GraphQL mutation via rsm2-api",
        "orgId": "4fe9b685-4451-48d7-b3f2-f79bd5553638",
        "userField1": "LET_LEARNER_SELECT"
    }
}
```

Retrieve all users with attributes matching any of the supplied email, first, and last name.

Perform searchUser with the three variables and return specific user attributes. If the user is not in a group, the orgId should be used. Otherwise, the groupId of the associated group must be used.

Call

```
query userSearchQuery($groupId: ID!, $firstName: String!, $lastName:
String!, $email: String!) {
  userSearch(
    groupId: $groupId
    firstName: $firstName
    lastName: $lastName
    email: $email
  ) { id firstName lastName email}
}
```

Response


```
{
    "data": {
        "userSearch": [
            {
                "id": "0824c08a-a016-49ab-9b78-3069b2981353",
                "firstName": "Livana",
                "lastName": "Aiello",
                "email": "liv-aiell@autozone-inc.info"
            },
            {
                "id": "003766ae-acda-4148-9784-23db8de8a88f",
                "firstName": "Randie",
                "lastName": "Aiello",
                "email": "randi.aiel@diaperstack.com"
            }
        ]
    }
}
```


Variables:


```
{"groupId": "202af8f8-fdd3-4bd9-8ff4-68d034d08831", "firstName": "Livana", "lastName": "Aiello", "email": "liv-aiell@autozone-inc.info"}
```


Batch operations

Several graphql queries or mutations can be batched in a single POST request. i.e. create or delete multiple users.

Here is a batch delete example, deleting 2 users in cURL:

```
curl 'https://rsm2-api.rosettastone.com/graphql' -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'Authorization: Bearer
ef726738-fb0b-44ac-9e11-48e4cf8775b0' --data-binary '[{"query":"mutation deleteUserMutation($input: DeleteUserInput!){\n  deleteUser(input:
$input) {org {id name namespace}}\n}","variables":{"input":{"userId":"b77425c9-1c61-45cd-b1b8-184e3715384a"}}}, {"query":"mutation
deleteUserMutation($input: DeleteUserInput!){\n  deleteUser(input: $input) {org {id name namespace}}\n}","variables":{"input":{"userId":"57b8b856-
cc2c-49cc-8abc-a38096859b07"}}}]' --compressed
```

Fluency Builder API

Using API methods

The Fluency Builder API utilizes GraphQL. While typical REST APIs require loading from multiple URLs, GraphQL APIs get all the data needed in a single request. The Fluency Builder API methods include:

- Query
  - Getting course information
  - Getting assigned courses to a user
- Mutation
  - Assigning courses to users
  - Deleting (unassigning) courses from users

In reference to schema, these queries and mutations can be formed to retrieve only the information required.

Conceptual Overview

- A Course is a collection of Lessons.
  - Each Lesson is a collection of Activities
- Assigned Courses are shown on the Fluency Builder home page
- Courses must be assigned before that content can be accessed. A URL deep linking into course content will fail if that course is not assigned

A list of courses can be downloaded here.

Example queries and mutations

assignedCourses Returns all courses(goals) assigned to a specific user.

Request

```
query{
  assignedCourses(userId: "$userId") {
    courseId,
    cefr,
    title,
    description
  }
}
```

Results

```
"data": {
  "assignedCourses": [
    {
      "courseId": "d5ff4823b4d770eab43f47fd7e1fb139",
      "cefr": "B1",
      "title": "Leading Meetings and Presentations (B1)",
      "description": "Le programme de formation Réunion et présentation va vous permettre d'avoir des échanges ..."
    },
    ...
    ]
  }
}
```

Variables

```
{“userId”: “e4514c64-0e31-439a-ab59-2c95a23795ef”}
```

Errors: code: ER_DUP_ENTRY is returned when attempting to add a course already assigned to that user

assignCourse Assigns one course to one user.

Request

```
mutation ($userId: String, $courseId: String!, $cefr: String!){
  assignCourse(userId: $userId, courseId: $courseId, cefr: $cefr) {
    courseId
    cefr
    description
  }
}
```

Results

```
"data": {
  "assignCourse": [
    {
      "courseId": "<assignedCourseId1>"
    },

    ...
    ]
  }
}
```

Variables

```
{
"userId": "e4514c64-0e31-439a-ab59-2c95a23795ef",
"courseId": "e8ec0a16bc43d1f756b5f473b5058c66",
"cefr": "B1"
}
```

deleteCourses Unassigns a course from a user.

Request

```
mutation {
deleteCourses(userId:"e4514c64-0e31-439a-ab59-2c95a23795ef",
courseId: "d6a7be9b7e952803d794bc0f49b13759") {
    courseId
  }
}
```

Results

```
"data": {
  "deleteCourses": [
    {
      "courseId": "<assignedCourseId1>"
    },
    ...
  ]
  }
}
```

Query the Course Progress

Fluency Builder Course Progress

Note: This query mirrors the content of the Fluency Builder report that is available in the Administrator Tools
This cURL allows the extraction of the progress made in each Fluency Builder course and lesson.

```
curl -H "Accept: application/json" -H "Authorization: Bearer fakeToken-a753-439b-ae40-713cbb855138"
"https://prism.rosettastone.com/reports/gaia/userActivity/fakeUserId-ac01-4f39-b477-e2d87ff36257?productId=product.6a1a08c9-7e8d-c5cc-58a8-96e93a35df94&startTimestamp=2021-06-16T00:00:00Z&endTimestamp=2022-01-01T00:00:00Z"
```

Where:
productId = product.6a1a08c9-7e8d-c5cc-58a8-96e93a35df94 (Fixed, this is the Fluency Builder product identifier)
userId = fakeUserId-ac01-4f39-b477-e2d87ff36257 (variable, replace with the ID of the user that you want to run the extraction for)
startTimestamp = 2021-06-16T00:00:00Z (variable, start date and time of the period that you want to report)

endTimestamp = 2022-01-01T00:00:00Z (variable, end date and time of the period that you want to report)

Foundations Course Progress

Note: This query mirrors the Learner Progress Report content that is available in the Administrator Tools
This cURL allows the extraction of the progress made in each Foundations Lesson.

```
curl -H "Accept: application/json" -H "Authorization: Bearer 4119ba88-a753-439b-ae40-713cbb855138"
"https://prism.rosettastone.com/reports/rsclassic/progress/fakeUserId-cf16-4188-94d7-
4b0d2640c7a9?async=false&languageLevelId=10&organizationId=fakeClientId-0df6-49f8-bdfb-96889bbcbf1d&product=product.d353e204-5952-4175-be5e-
0a5630bfab6c&start=2022-02-01&end=2022-04-05"
```

Where:
fakeUserId-cf16-4188-94d7-4b0d2640c7a9 (user ID of the learner that you want to query)
async=false (wait for the query results)
languageLevelId=10 (variable, ID of the language level that you want to query. 10 is for English American Level 1. See appendix for the entire list)
organizationId=fakeClientId-0df6-49f8-bdfb-96889bbcbf1d (Fixed, unique to each client)
product=product.d353e204-5952-4175-be5e-0a5630bfab6c (Fixed, Rosetta Stone Foundations identifier)
start=2022-02-01 (variable, start date of the period that you want to report)
end=2022-04-05 (variable, end date of the period that you want to report)

Troubleshooting

“Variable \” “\ is never used”

- Check the formatting of variables passed. The structure may be incorrect or additional and unexpected parameters are included such as “input”.
- Check query variable declaration types. Refer to the schema for appropriate type i.e. “String”, “String!”, “Int”, “Boolean”.
- Check variable declaration text in the query to match variable labels.

Additional Resources

Much of the RSM2-api was modelled after the GitHub v4 api.

- More on forming queries and mutations
- GraphQL Catalyst API schema

Appendix I: Roseta Stone Catalyst Languages

id	name	IETF code
1	Arabic Level 1 (Units 1-4)	ar
2	Arabic Level 2 (Units 5-8)	ar
3	Arabic Level 3 (Units 9-12)	ar
4	German Level 1 (Units 1-4)	de-DE
5	German Level 2 (Units 5-8)	de-DE
6	German Level 3 (Units 9-12)	de-DE
7	English (British) Level 1 (Units 1-4)	en-GB
8	English (British) Level 2 (Units 5-8)	en-GB
9	English (British) Level 3 (Units 9-12)	en-GB
10	English (American) Level 1 (Units 1-4)	en-US
11	English (American) Level 2 (Units 5-8)	en-US
12	English (American) Level 3 (Units 9-12)	en-US
13	Spanish (Spain) Level 1 (Units 1-4)	es-ES
14	Spanish (Spain) Level 2 (Units 5-8)	es-ES
15	Spanish (Spain) Level 3 (Units 9-12)	es-ES
16	Spanish (Latin America) Level 1 (Units 1-4)	es-419
17	Spanish (Latin America) Level 2 (Units 5-8)	es-419
18	Spanish (Latin America) Level 3 (Units 9-12)	es-419
19	French Level 1 (Units 1-4)	fr-FR
20	French Level 2 (Units 5-8)	fr-FR
21	French Level 3 (Units 9-12)	fr-FR
22	Italian Level 1 (Units 1-4)	it-IT
23	Italian Level 2 (Units 5-8)	it-IT
24	Italian Level 3 (Units 9-12)	it-IT
25	Portuguese (Brazil) Level 1 (Units 1-4)	pt-BR
26	Portuguese (Brazil) Level 2 (Units 5-8)	pt-BR
27	Portuguese (Brazil) Level 3 (Units 9-12)	pt-BR
28	Russian Level 1 (Units 1-4)	ru-RU
29	Russian Level 2 (Units 5-8)	ru-RU
30	Russian Level 3 (Units 9-12)	ru-RU
33	Chinese (Mandarin) Level 1 (Units 1-4)	zh-CN
34	Chinese (Mandarin) Level 2 (Units 5-8)	zh-CN
35	Chinese (Mandarin) Level 3 (Units 9-12)	zh-CN
36	Irish Level 1 (Units 1-4)	ga-IE
37	Irish Level 2 (Units 5-8)	ga-IE
38	Irish Level 3 (Units 9-12)	ga-IE
39	Japanese Level 1 (Units 1-4)	ja-JP
40	Japanese Level 2 (Units 5-8)	ja-JP
41	Japanese Level 3 (Units 9-12)	ja-JP
42	Hebrew Level 1 (Units 1-4)	he-IL
43	Hebrew Level 2 (Units 5-8)	he-IL
44	Hebrew Level 3 (Units 9-12)	he-IL
45	Persian (Farsi) Level 1 (Units 1-4)	fa-IR
46	Persian (Farsi) Level 2 (Units 5-8)	fa-IR
47	Persian (Farsi) Level 3 (Units 9-12)	fa-IR
48	Hindi Level 1 (Units 1-4)	hi-IN
49	Hindi Level 2 (Units 5-8)	hi-IN
50	Hindi Level 3 (Units 9-12)	hi-IN
51	Polish Level 1 (Units 1-4)	pl-PL
52	Polish Level 2 (Units 5-8)	pl-PL
53	Polish Level 3 (Units 9-12)	pl-PL
54	Greek Level 1 (Units 1-4)	el-GR
55	Greek Level 2 (Units 5-8)	el-GR
56	Greek Level 3 (Units 9-12)	el-GR
57	Korean Level 1 (Units 1-4)	ko-KR
58	Korean Level 2 (Units 5-8)	ko-KR
59	Korean Level 3 (Units 9-12)	ko-KR
60	Dutch Level 1 (Units 1-4)	nl-NL
61	Dutch Level 2 (Units 5-8)	nl-NL
62	Dutch Level 3 (Units 9-12)	nl-NL
63	Swedish Level 1 (Units 1-4)	sv-SE
64	Swedish Level 2 (Units 5-8)	sv-SE
65	Swedish Level 3 (Units 9-12)	sv-SE
66	Turkish Level 1 (Units 1-4)	tr-TR
67	Turkish Level 2 (Units 5-8)	tr-TR
68	Turkish Level 3 (Units 9-12)	tr-TR
69	Vietnamese Level 1 (Units 1-4)	vi-VN

70	Vietnamese Level 2 (Units 5-8)	vi-VN	
71	Vietnamese Level 3 (Units 9-12)	vi-VN	
75	Filipino (Tagalog) Level 1 (Units 1-4)	tl-PH	
76	Filipino (Tagalog) Level 2 (Units 5-8)	tl-PH	
77	Filipino (Tagalog) Level 3 (Units 9-12)	tl-PH	
78	English (American) Level 4 (Units 13-16)		en-US
79	English (American) Level 5 (Units 17-20)		en-US
80	Spanish (Latin America) Level 4 (Units 13-16)		es-419
81	Spanish (Latin America) Level 5 (Units 17-20)		es-419
82	Spanish (Spain) Level 4 (Units 13-16)	es-ES	
83	Spanish (Spain) Level 5 (Units 17-20)	es-ES	
84	French Level 4 (Units 13-16)	fr-FR	
85	French Level 5 (Units 17-20)	fr-FR	
86	Italian Level 4 (Units 13-16)	it-IT	
87	Italian Level 5 (Units 17-20)	it-IT	
88	German Level 4 (Units 13-16)	de-DE	
89	German Level 5 (Units 17-20)	de-DE	
102	English (British) Level 4 (Units 13-16)	en-GB	
103	English (British) Level 5 (Units 17-20)	en-GB	
113	Russian Level 4 (Units 13-16)	ru-RU	
114	Russian Level 5 (Units 17-20)	ru-RU	
115	Chinese (Mandarin) Level 4 (Units 13-16)		zh-CN
116	Chinese (Mandarin) Level 5 (Units 17-20)		zh-CN