Getting Started for ADF App Developers

Introduction

The Singapore Student Learning Space (SLS) is a learning portal to support new ways of learning for students.

In the SLS, students and teachers interact in the Virtual Learning Environment (VLE). The Application Development Framework (ADF) was conceptualised to allow the VLE to be extended with external, specialised tools. This is achieved by allowing the integration of external web applications with the SLS, with data exchange facilitated through API.

The ADF is developed using GraphQL and will be able to support Learning Tool Interoperability (LTI) v1.3 in the future. For more info on LTI, visit http://www.imsglobal.org/spec/lti/v1p3/

In this document, we share a sample use case, and the technical documentation for integration with the ADF. The ADF currently only supports the integration of web applications that are maintained independently by third-party developers.

Key Concepts

Each ADF app is:

- A web application with a web server to make API calls to SLS;
- Allowed to be installed into group(s) that is either a class taking a subject in a level (e.g. Secondary 2 Biology) or a group created in SLS without a subject/ level;
- Able to make API calls using a valid Access Token (e.g. get information about groups that have the app installed);
- Launched by users from SLS with the context of a VLE group;
- Displayed either in a new tab or iframe within the SLS VLE application when launched.

Sample Use Case Walkthrough

For a sample use case of how an app is integrated into SLS through ADF, please refer to Sample Walkthrough.

ADF APIs

API Versioning

Semantic versioning is used to version the APIs. Breaking changes that are not backwards compatible will be released in major versions, while backwards compatible modifications would be released in minor versions.

API Specifications

For API authentication, please refer to ADF API Authentication v1.2.

For application integration, please refer to ADF Data API Specifications v1.4.

What's Next

Please contact us to discuss potential integration opportunities. We provide a few tools to help with integration:

- Sample credentials for testing ADF APIs (clientId and clientSecret);
- Sample SLS accounts to view VLE as a student and teacher;
- GraphiQL for developers to test GraphQL queries against ADF.

Sample Use Case Walkthrough

Introduction

This walkthrough will provide a sample use case of how an app can be integrated, and how some of the APIs can be used. In this example, the integration of Application_1 will be explored.

Application_1

Application_1 is a Learning Management System (web application) which can create programming quizzes. Students are able to enter code (eg. Python or Java), which will be evaluated and run against test cases set by the teacher to determine the correctness of the code/function. This example will demonstrate one way in which Application_1 can be integrated with ADF, in which each programming quiz is mapped to a SLS Assignment.

Sample Sequence

This is a sample sequence of assigning and completion of a Application_1 quiz. Assume that Application_1 is installed into a Subject Group (Sec 4E1 Computing).

Teacher (Ms Lim)

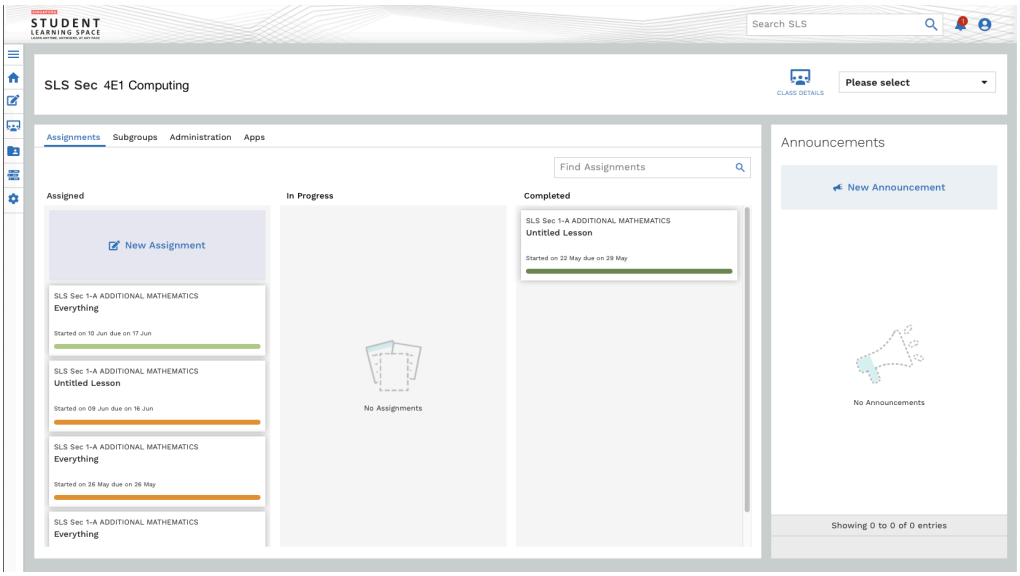
1. In Sec 4E1 Computing of the SLS VLE interface, Ms Lim launches a Application_1 Programming Quiz.
2. Application_1 is launched within an iFrame in SLS.
    1. (*GET Context*) During launch, SLS would provide Application_1 with a *context-id* through the launch URL (eg. *Application_1.com/launch?context-id=1234*).
        1. Application_1 makes an API call to get information on the *context-id*, including the current user and group which is launching Application_1.
        2. The information of the event 'LAUNCH_APP', and the event_id would be the UUID of the group.
    2. (*Query Group*) If the group is new, Application_1 can make an API call to get the information of the group.
    3. Application_1 will be redirected to the landing page, signed in as Ms Lim.
3. Ms Lim creates a programming quiz on Application_1, and assigns it to the group.
    1. (*Mutation CreateAssignment*) Application_1 can make an API call to add an assignment to SLS, and provide the list of students it is assigned to.
    2. SLS would create the necessary Assignment and Tasks (based on the assignees provided by the mutation), and respond with the necessary information.
4. Ms Lim can launch this programming quiz through the SLS Assignments page. This is meant for Ms Lim to view, edit and monitor the assignment in Application_1.
    1. (*GET Context*) During launch, SLS would also provide Application_1 with a *context-id.*
        1. The information of the event 'LAUNCH_ASSIGNMENT', and the event_id would be the UUID of the Assignment.
    2. Application_1 will be redirected to a page to view/edit/monitor the assignment, signed in as Ms Lim.

Student (John)

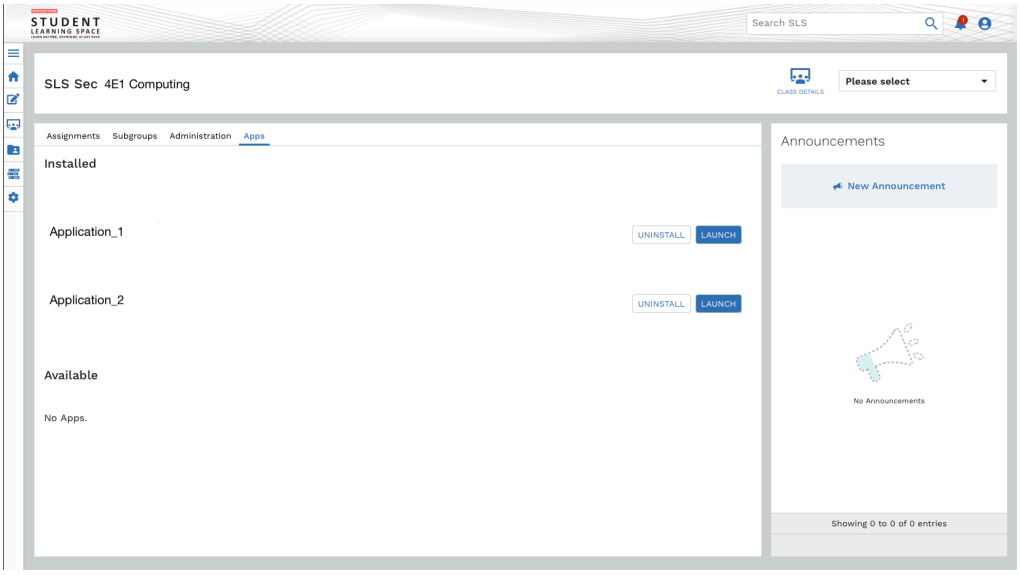
1. When the Assignment has started, the Programming Assignment as a Task can be seen by John on their SLS space. Launching the task would bring them into Application_1:
    1. (*GET Context*) During launch, SLS would also provide Application_1 with a *context-id.*
        1. The information of the event 'LAUNCH_TASK', and the event_id would be the UUID of the Task.
    2. Application_1 will be redirected to a page to view/edit the task, signed in as John.
2. John will start work on the programming quiz in Application_1.
    1. (*Mutation UpdateTask*) (Optional) If preferred, Application_1 can make a call to update the status of the task from 'NEW' to 'IN_PROGRESS'.
    2. (*Mutation UpdateTask*) Upon completion, Application_1 can make a call to update the status of the task to 'COMPLETED'
3. John will be still able to launch Application_1 Tasks that are 'IN_PROGRESS' or 'COMPLETED', and will be launched within an iFrame in SLS.
    1. (*GET Context*) During launch, SLS would provide Application_1 with a *context-id* through the launch URL (eg. *Application_1.com/launch?context-id=1234*).
        1. Application_1 makes an API call to get information on the *context-id*, including the current user who is launching a task in Application_1.
        2. The information of the event 'LAUNCH_TASK', and the event_id would be the UUID of the Task.

Sample Screens

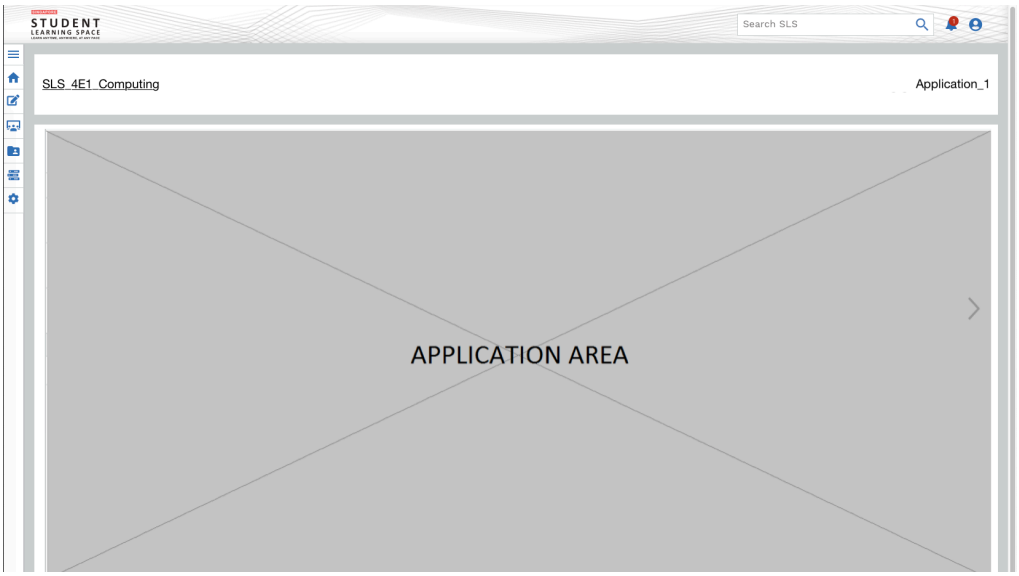
Click the Apps tab to access the applications.



A menu with the installed and available applications will show up.



Clicking on the launch icon will launch the application (either within an iFrame, or in a new tab)



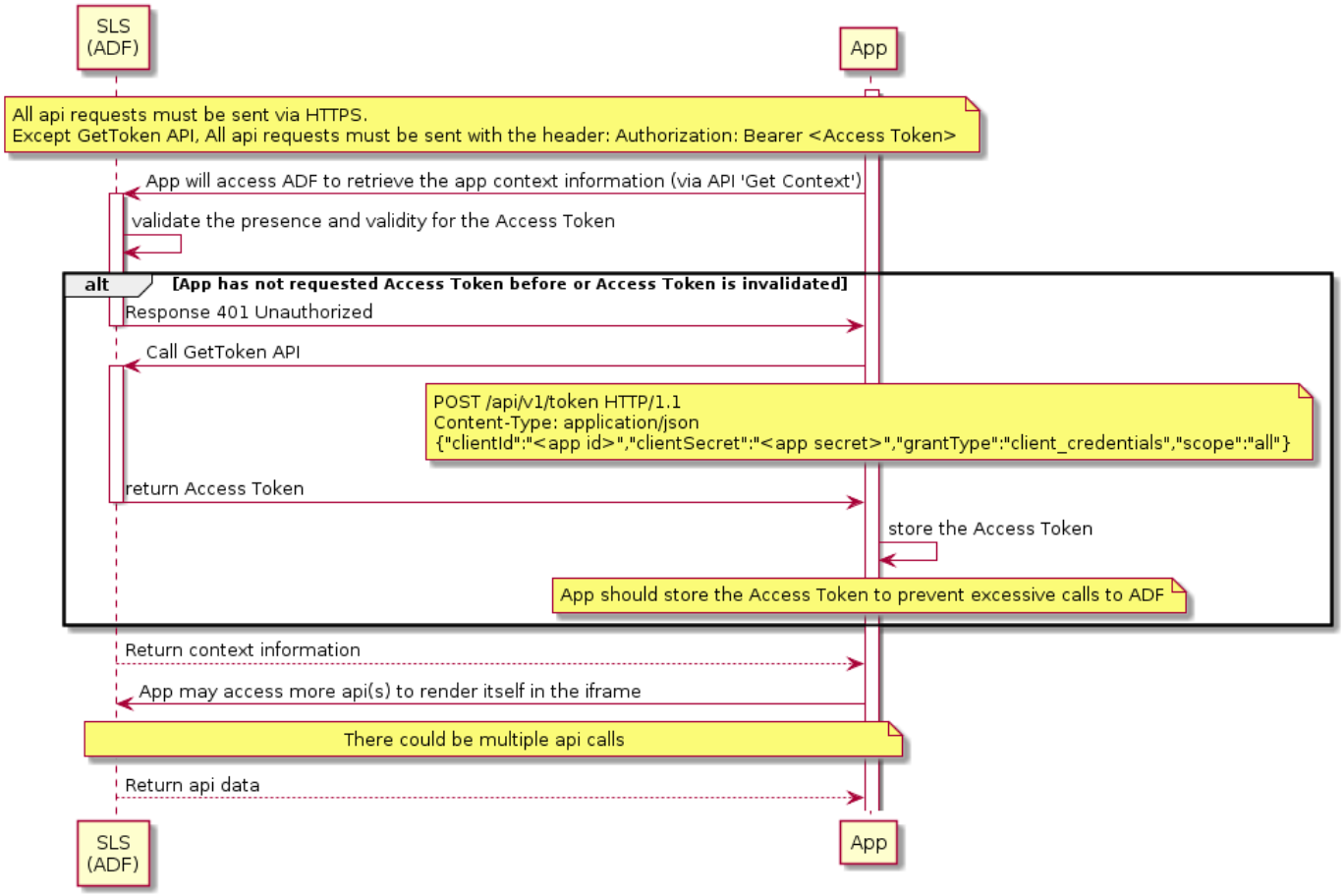
ADF API Authentication v1.2

This document is a working draft; feel free to contact us for any issues listed.

Overview

This document explains how to set up an application to make API calls with the ADF API authentication mechanism.

For API communication and authentication, a 2-leg OAuth authentication is used (or Client credentials grant type for Oauth 2.0). Using the provided clientId and clientSecret, an application can obtain an authorization token via the Get Access Token endpoint. This token can be used to make API calls to retrieve data from the APIs.



Get Access Token (REST)

Get access token to access other API methods.

URL

```
POST /apis/v1/token
```

Parameters

Type	Parameter	Description	Mandatory	Default	Remarks
string	clientId	App identifier which assigned by SLS	Yes		
string	clientSecret	Shared secret between App and SLS which is generated by SLS	Yes		
string	grantType	Value MUST be set to "client_credentials".	Yes		
string	scope	The scope of the access token issued	No	all	Reserved for future use

Sample Request

```
{"clientId":"f956dd99-wdbc-11e7-a92d-0qw021fa327c","clientSecret":"
abcdefgh","grantType":"client_credentials","scope":"all"}
```

Sample Response

```
{
        "token": "ee5d40b0-1fdn-4975-ab07-44b410f210c2"
}
```

Error Responses

HTTP Code	Description	Response
200	App is not registered or not approved.	{"errors":[{"message":"Invalid credential (40001)"}]}
200	App secret is wrong.	{"errors":[{"message":"Invalid credential (40002)"}]}
others (e.g. 4xx, 5xx)	Call failed due to either client-side problem or server-side problem. Refer to List_of_HTTP_status_codes for more details.	

ADF Data API Specifications v1.4

This document is a working draft; feel free to contact us for any issues listed.

Overview

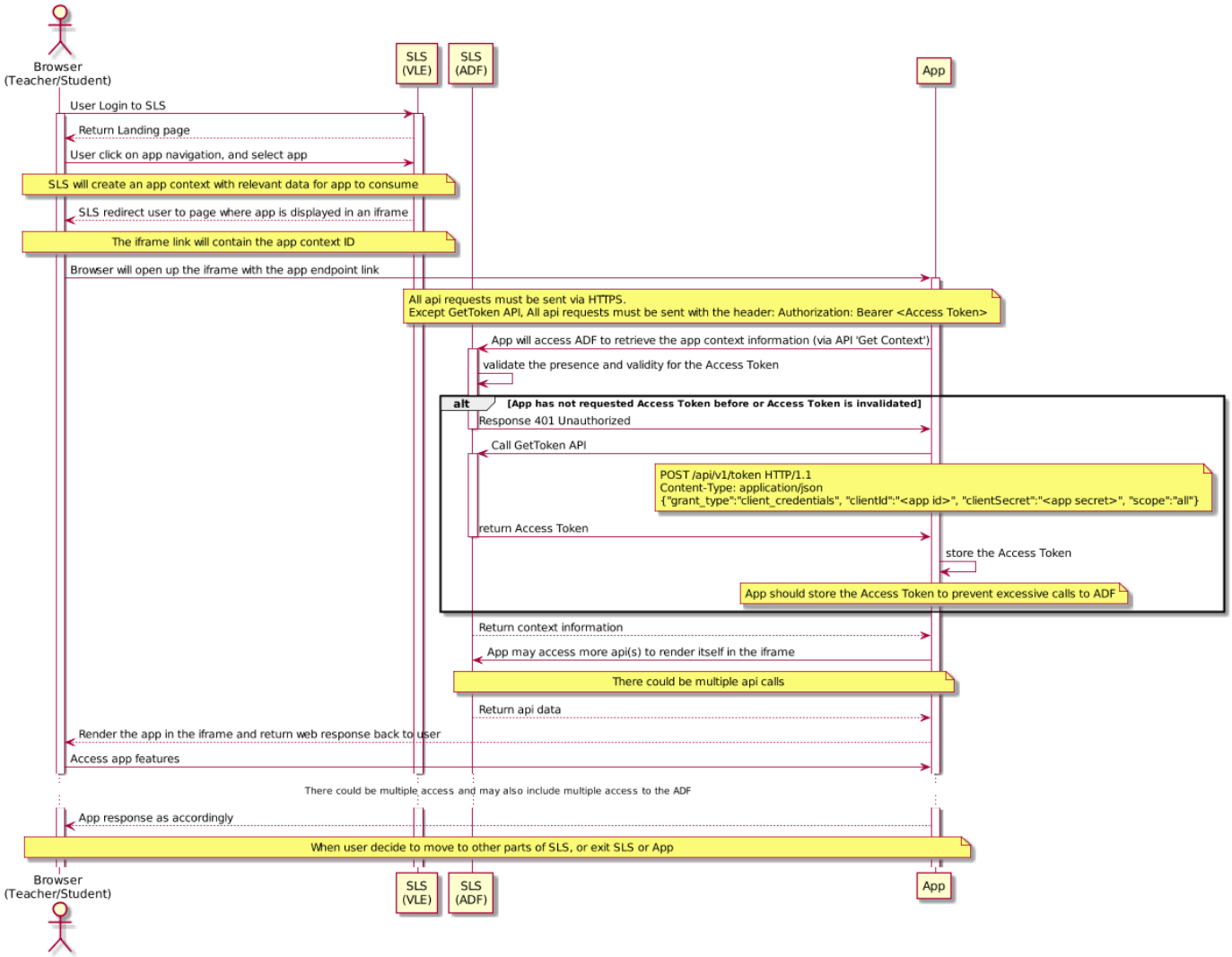
The following are the API specifications for Application Integration. The APIs follow GraphQL; if you are new, you can learn more through https://www.howtographql.com/.

In this specification, we provide the following information:

1. App Launch Flow
2. The GraphQL API Endpoint
3. Invoking and Testing GraphQL Calls
4. GraphQL Resource Limitations
5. Pagination
6. GraphQL Schema (Scalars, Objects, Queries and Mutations)
7. GraphQL Queries and Mutations

App Launch Flow

This is a sample sequence when an app is launched



The GraphQL API Endpoint

The GraphQL API has a single endpoint, which remains constant regardless of the operation

Invoking and Testing GraphQL Calls

To make an API call with GraphQL:

- Use HTTP POST to make the API call
- In the request header, specify the following:
    - authorization: Bearer <Access Token>
    - content-type: application/json
- In the request body, include the GraphQL query in a string as part of a JSON object.

It is recommended using the GraphiQL Explorer to test GraphQL calls. You can also use cURL or any other library to make HTTP calls. To query GraphQL using cURL, make a `POST` request with a JSON payload. The payload must contain a string called `query`:

```
curl -H "Authorization: Bearer <Access Token>" -X POST -d " \
 { \
   \"query\": \"query { context(uuid:\"f6fc25d6-fee1-45b5-b149-
d669c4d182cf\") { user {id} }}\" \
 } \
"
```

GraphQL Resource Limitations

To protect the ADF server from excessive and abusive queries, there are resource limitations set on the server. Currently, there are two limitations introduced on the ADF server:

1. Maximum query depth (set at 3); and
2. Query complexity (with an arbitrary high limit set for now)

A more formalised version of the resource limitations will be implemented and documented soon. The current set limitations should prevent abusive GraphQL calls and not affect most queries; if you face issues, please reach out to the SLS ADF team.

GraphQL Pagination

ADF follows the standard GraphQL cursor-based pagination, which supports client to

- Paginate through the list
- Ask for information about the connection itself, like totalCount or pageInfo.
    - Client should always query for the pageInfo to know whether next page is available, or whether this is the last page.

Pagination support is currently available in allGroups Query.

GraphQL Schema

Scalars

```
# Date and time format, format: ISO_OFFSET_DATE_TIME, e.g.,2007-12-03T10:
15:30+08:00
DateTime
# Unique identifier
UUID
```

Object

PageInfo

Cursor based pagination

```
type PageInfo {
  # The cursor value from the last node in the returned page.
  endCursor: String!
  # Whether there is next page available.
  hasNextPage: Boolean
}
```

User

```
# A user in the system, who is either a teacher or student.
type User {
        # Login Id of user, unique identifier of user.
        uuid: UUID!

        # Name of user.
        name: String!

        # Role of user, either Teacher or Student.
        role: Role!

        # School of user
         school: School

        # Email address of user
        # Not all users have email addresses
    # Prior approval is required to call this field
        email: String

        # Level that student is in, e.g. Secondary 1. Applicable to
student only (Null for teacher).
        level: Level

        # The unique number issued to each student in his/her class.
Applicable to student only (Null for teacher).
        classSerialNo: Int

        # Active (ie. current year) groups that user is in.
        # Only returns groups that have this app installed.
        # For Teacher, it is the Groups that this teacher is teaching
        # For Student, it is the Groups to which this student belongs
        #
    # first: returns the first N groups, default to 20 if not specified.
         groups(first: Int = 20): [Group!]
}

enum Role {
        # Teacher
        TEACHER

        # Student
        STUDENT
}


enum Level {
  # Primary 1
  PRIMARY_1
```

```
  # Primary 2
  PRIMARY_2

  # Primary 3
  PRIMARY_3

  # Primary 4
  PRIMARY_4

  # Primary 5
  PRIMARY_5

  # Primary 6
  PRIMARY_6

  # Secondary 1
  SECONDARY_1

  # Secondary 2
  SECONDARY_2

  # Secondary 3
  SECONDARY_3

  # Secondary 4
  SECONDARY_4

  # Secondary 5
  SECONDARY_5

  # Pre-U 1
  PRE_U_1

  # Pre-U 2
  PRE_U_2

  # Pre-U 3
  PRE_U_3
}
```

Group

```
# A group of teachers and students
type Group {
        # UUID of the Group.
        uuid: UUID!

        # Name of the Group used in VLE.
        name: String!

        # Identifier of Group for students and teachers, e.g. S1-A.
```

```
        code: String

        # Subject of the Group: e.g. English Language.
        subject: String

        # School of the Group
        school: School

        # Whether the Group is currently active.
        active: Boolean!

        # Level of the group: e.g. Secondary 1.
        level: Level

        # Teacher(s) who teaches this Group.
        # first: returns the first N teachers, default to 0 if not
specified.
        teachers(first: Int = 0): [User!]!

        # Students who belong to this Group.
        #
    # first: returns the first N students, default to 0 if not specified.
    # sortField: sorts students by the field, default to class serial no.
if not specified.
        students(first: Int = 0, sortField: StudentSortBy =
"CLASS_SERIAL_NO"): [User!]!

        # Assignments that have been assigned to this Group.
        #
    # first: returns the first N assignments, default to 20 if not
specified.
    assignments(first: Int = 20): [Assignment!]!

    # The date time when the Group last changed.
    # The field is updated when Group name, membership or status is changed
    lastUpdated: DateTime!
}


enum StudentSortBy {
        # Sort students by class serial number
        CLASS_SERIAL_NO

        # Sort students by name
        NAME
}


type GroupsConnection {
  pageInfo: PageInfo!
  edges: [GroupsEdge]!
}
```

```
type GroupsEdge {
  cursor: String!
  node: Group!
}
```

School

```
type School {
        # School code
        code: String!

        # The full name of school
         name: String!
}
```

Context

```
# A context is provided to an App when a user launches the App from SLS.
# The context provides information of the user who launches the App and
from where it is launched.
type Context {
        # Information of the user who launches the App
        user: User!

        # Information of where the App is launched
    event: Event!
}

# Event information when App is launched.
# Event is generated when:
#   1. LAUNCH_APP: Launch App from <Group> page
#   2. LAUNCH_ASSIGNMENT: Open <Assignment> page (from Assignment Listing
page)
#   3. LAUNCH_TASK: Open <Task> page (from either Dashboard or
Notification)
#        4. LAUNCH_URL: Launch a URL embedded in lesson
type Event {
  # The type of event associated with the context.
  type: EventType!

  # UUID associated with the EventType, ie. Group / Assignment / Task.
Empty for LAUNCH_URL
  typeId: UUID

  # The launchKey of event associated with the context.
  launchKey: String

  # The launchUrl of event associated with the context.
```

```
  launchUrl: String
}

enum EventType {
  # Launch application.
  LAUNCH_APP

  # Launch assignment. E.g. Teacher open assignment.
  LAUNCH_ASSIGNMENT

  # Launch task. E.g. Student open task from dashboard.
  LAUNCH_TASK

  # Launch a url embedded in Lesson.
  LAUNCH_URL
}
```

Assignment

```
# A piece of work that a teacher allocates/assigns to one or more students
in a Group.
# Each assignment has a collection of tasks, each corresponding to a
student in the assignment.
# It is not mandatory for all students in the Group to have a task for
each assignment.
type Assignment {
        # UUID of this assignment.
        uuid: UUID!

        # Title of assignment.
        title: String!

        # Assignment start date and time, format: ISO_OFFSET_DATE_TIME, e.
g.,2007-12-03T10:15:30+01:00.
        start: DateTime!

        # Assignment end date and time, format: ISO_OFFSET_DATE_TIME, e.g.,
2007-12-03T10:15:30+01:00.
        end: DateTime

        # User (ie. a teacher) who has created this assignment.
        createdBy: User!

        # User (ie. a teacher) who have made changes to this assignment.
        modifiedBy: User

    # Type of assignment, reflected in the SLS assignment page.
    type: AssignmentResourceType!

        # Group that this resource has been assigned to.
        group: Group!
```

```
        # List of tasks that have been generated.
        #
        # first: returns the first N tasks, default to 0 if not specified.
        tasks(first: Int = 0): [Task]!

        # Whether the assignment will launch in a new tab or within SLS's
frame
        # True if new tab, False if otherwise.
        openInNewTab: Boolean
}

enum AssignmentResourceType {
    LESSON
    QUIZ
}
```

Task

```
# A basic unit of work for one student.
# Each task is part of an assignment, and tracks the progress of the
student's work.
type Task {
        # UUID of this task.
        uuid: UUID!

        # Title of assignment that this task is associated to.
        title: String!

        # Task start date and time (same as assignment), format:
ISO_OFFSET_DATE_TIME, e.g.,2007-12-03T10:15:30+08:00.
        start: DateTime!

        # Task end date and time (same as assignment), format:
ISO_OFFSET_DATE_TIME, e.g.,2007-12-03T10:15:30+08:00.
        end: DateTime

        # User (ie. teacher) who has created this task.
        createdBy: User!

        # User (ie. student) assigned to this task.
        assignee: User!

        # Subject of the task assigned.
        subject: String!

        # Status of this task, default is NEW.
        status: TaskStatus!
}

# Status of the task by the assigned student.
```

```
enum TaskStatus {
  # Student has not started on the task.
  NEW

  # Student has started, but not completed the task.
  IN_PROGRESS

  # Student has completed the task.
  COMPLETED
}
```

Notification

```
# A notification is a message for a specific SLS user, with a message to
provide information.
# The notification can be clickable if the eventType and eventTypeId is
provided.
# If both eventType and eventTypeId are not provided, the notification is
not clickable.
type Notification {
    # UUID of the notification.
    uuid: UUID!

          # Notification message.
          message: String!

        # Scope in which notification is sent in the context from.
Examples include global, school or group.
        # This is for future extensibility and GROUP should be selected.
        scope: NotificationScope

        # Id of scope (for NotificationScope of GROUP, this should be set
to Group UUID).
        scopeId: String

          # Context event type set in the Context when the user clicks and
launches the notification.
          # eventType must be one of the enum values of EventType.
          eventType: String

          # Context event type id set in the Context when the user clicks
and launches the notification
          eventTypeId: UUID

          # List of the recipient's (ie. SLS user) UUID.
    # Recipient(s) must exists within the scope (one of the
NotificationScope values), specified by the scopeId.
          recipient:[UUID!]!

        # Whether the notification, when clicked, will launch in a new tab
or within SLS's frame
```

```
        # True if new tab, False if otherwise.
        openInNewTab: Boolean
}

enum NotificationScope {
        GROUP
    SCHOOL
        GLOBAL
}
```

Query

```
type Query {

  # Retrieve context information via Context ID which is provided by SLS
  # Various Scenarios:
  #   1. LAUNCH_APP: Launch App from <Group> page
  #   2. LAUNCH_ASSIGNMENT: Open <Assignment> page (from Assignment
Listing page)
  #   3. LAUNCH_TASK: Open <Task> page (from either Dashboard or
Notification)
  #
  # Context will expire in 10 seconds.
  #
  # Errors:
  #   1. Authorization header is invalid
  #   2. Context does not exist
  #   3. Context has expired
  #
  context(uuid: UUID!): Context

  # Retrieve user information via user ID.
  #
  # Errors:
  #   1. Authorization header is invalid
  #   2. User does not exist
  #   3. User is neither Teacher nor Student
  #   4. App is not installed for this User
  #
  user(uuid: UUID!): User

  # Retrieve group information via group UUID.
  #
  # Errors:
  #   1. Authorization header is invalid
  #   2. Group does not exist
  #   3. App is not installed for this group
  #
  group(uuid: UUID!): Group
```

```
  # Retrieve assignment information via the assignment UUID.
  #
  # Errors:
  #   1. Authorization header is invalid
  #   2. Assignment does not exist
  #   3. Assignment is not created from this App
  #
  assignment(uuid: UUID!): Assignment

  # Retrieve task information via the task UUID.
  #
  # Errors:
  #   1. Authorization header is invalid
  #   2. Task does not exist
  #   3. Task is not created from this App
  #
  task(uuid: UUID!): Task

  # Retrieve all groups installed to the App.
  #
  # If changedSince is provided, only groups that changed since the time
provided will be returned.
  # If schoolCode is provided, only groups in the school will be returned.
  #
  # Errors:
  #   1. Authorization header is invalid
  #   2. Timestamp is invalid
  #   3. School code is invalid
  #
  allGroups(changedSince: DateTime, schoolCode: String, first: Int, after:
String): GroupsConnection
}
```

Mutation

Convention:

1. For create and update of Types, to return the Type as part of the response.
2. For deletion of Types, the unique identifier (ie. UUID) of the Type is returned as part of the response.

```
type Mutation {
        # Creates an assignment for a group.
        # To assign the same quiz/activity/assessment to multiple Groups,
create an assignment for each group.
        #
        # Errors:
        #        1. Authorization header is invalid.
        #         2. Assignment start date is after or equal end date.
        #         3. Assignment title is not provided.
        #         4. Assignee is empty.
        #         5. Group is not provided.
        createAssignment(input: AssignmentInput!): Assignment
```

```
        # Updates the assignment: set fields to null if there are no
changes to those assignment fields
        # Use this mutation to add or remove tasks associated to this
assignment.
        #
        # Errors:
        #       1. Authorization header is invalid.
        #       2. Assignment UUID is invalid.
        #       3. Assignment does not belong to App.
    #   4. Once assignment has started, unable to update openInNewTab
field.
        updateAssignment(uuid: UUID!, input: AssignmentInput!): Assignment

    # Delete assignment: assignment UUID is returned on successful
deletion.
        #
        # Errors:
        #       1. Authorization header is invalid.
        #       2. Assignment UUID is invalid.
        #       3. Assignment does not belong to App.
    deleteAssignment(uuid: UUID!): UUID

        # Update task status to reflect the task progress.
    #
    # Errors:
        #       1. Authorization header is invalid.
        #       2. Task UUID is invalid.
        #       3. Task does not belong to App.
        updateTask(id: UUID!, status: TaskStatus!): Task

        # Create notification for SLS user.
    #
    # Errors:
        #       1. Authorization header is invalid.
        #       2. groupUuid is invalid. App can send notification only
to group(s) that has the App installed
        #       3. Recipient(s) does not exists or belong to the group.
        createNotification(input:NotificationInput!): Notification
}

# Input type for mutations associated with assignment.
input AssignmentInput {
          # Title of assignment.
          title: String

          # Assignment start date, format: ISO_OFFSET_DATE_TIME, e.g.,2007-
12-03T10:15:30+01:00
          start: DateTime

        # Assignment end date, format: ISO_OFFSET_DATE_TIME, e.g.,2007-12-
03T10:15:30+01:00
          end: DateTime
```

```
        # UUID of User (ie. a teacher) who has created this assignment.
        createdBy: UUID

        # UUID of User (ie. a teacher) who have made changes to this
assignment.
        modifiedBy: UUID

          # Type of assignment, reflected in the SLS assignment page.
    type: AssignmentResourceType

        # User UUIDs of students who are assigned to this assignment.
        # On successful creation of assignment, corresponding tasks for
each user will be created.
          assignees: [UUID!]

        # UUID of Group that the assignment is assigned to.
          groupUuid: UUID

        # When Student/Teacher launch this assignment, it will be opened
within SLS or new tab based on this setting.
        # If this value is not provided, it will default to application's
setting.
        # Once assignment has started, this value can no longer be updated
and error will be returned.
        openInNewTab: Boolean
}

# Input type to create notification.
input NotificationInput {
          # Notification message, maximum length of 300 characters.
          message: String!

         # Scope in which notification is sent in the context from.
Examples include global, school or group.
        # This is for future extensibility and GROUP should be selected.
        scope: NotificationScope

        # Id of scope (for NotificationScope of GROUP, this should be set
to Group UUID).
        scopeId: String

          # Context event type set in the Context when the user clicks and
launches the notification.
          # eventType must be one of the enum values of EventType.
          eventType: String

          # Context event type id set in the Context when the user clicks
and launches the notification
          eventTypeId: UUID

          # List of the recipient's (ie. SLS user) UUID.
    # Recipient(s) must exists within the scope (one of the
```

```
NotificationScope values), specified by the scopeId.
          recipient:[UUID!]!

        # When Student/Teacher launch this notification, it will be opened
within SLS (iframe) or new tab based on this setting.
        # If this value is not provided, it will default to application's
setting.
        openInNewTab: Boolean
}

enum NotificationScope {
    GROUP
    SCHOOL
    GLOBAL
}
```

Examples of GraphQL Queries and Mutations

The above schema should provide enough information to start integration. This is a more in-depth section exploring the queries and mutations for the ADF, which includes some explanation, sample request and responses.

Retrieve Context

Retrieve context information via Context ID which provided by SLS.

Notes

- To get access token, please refer to 'Get Access Token' API.
- A Context token is only valid for query within 10 seconds of issue, after which the query only returns an error that the context has expired.

Click here to expand...

URL

```
POST /apis/v1/graphql
```

Sample Request

```
{
  context(uuid:"f6fc25d6-fee1-45b5-b149-d669c4d182cf"){
    user{
      uuid
    }
    event{
      typeId
      type
    }
  }
}
```

Sample Response


```
{
 "data": {
   "context": {
     "user": {
       "uuid": "148e4c16-7b3b-11e8-adc0-fa7ae01bbebc"
     },
     "event": {
       "typeId": "90c63878-92e1-11e7-b105-0800279a327c",
       "type": "LAUNCH_APP"
     }
   }
 }
}
```


Error Responses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Requested context does not exist or doesn't belong to this app.	{ "data":{ "context":null }, "errors":[{ "message":"Exception while fetching data (/context) : Context does not exist" }] }
200	Requested context has expired.	{ "data":{ "context":null }, "errors":[{ "message":"Exception while fetching data (/context) : Context has expired" }] }

Retrieve User

Retrieve user information via user ID.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
{
  user(uuid:"26a8c4bc-7b3b-11e8-adc0-fa7ae01bbebc"){
    uuid
    name
    groups(first:2){
      uuid
      subject
      code
    }
  }
}
```

Sample Response

```
{
 "data": {
   "user": {
     "id": "26a8c4bc-7b3b-11e8-adc0-fa7ae01bbebc",
     "name": "Fay Nyeow",
     "groups": [
       {
         "uuid": "90c60139-92e1-11e7-b105-0800279a327c",
         "subject": "ADDITIONAL MATHEMATICS",
         "code": "S1-A"
       },
       {
         "uuid": "90c603db-92e1-11e7-b105-0800279a327c",
         "subject": "ADDITIONAL MATHEMATICS",
         "code": "S1-B"
       }
     ]
   }
  }
}
```

Error Responses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{

		"message":"Authorization header is invalid" }] }

200	User does not exist, user is neither teacher nor student, App is not installed to this User.	{
		"data":{
		"user":null
		},
		"errors":[
		{
		"message":"Exception while fetching data (/user) : User does not exist"
		}
		]
		}

Retrieve Group

Retrieve group information via group ID.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
{
  group(uuid:"90c6319d-92e1-11e7-b105-0800279a327c"){
    uuid
    code
    subject
    students(first:2, sortField:CLASS_SERIAL_NO){
      name
      uuid
    }
    teachers(first:2){
      name
      uuid
    }
  }
}
```

Sample Response

```
{
 "data": {
   "group": {
     "uuid": "90c6319d-92e1-11e7-b105-0800279a327c",
     "code": "S1-A",
     "subject": "TAMIL",
     "students": [
```

```
          {
            "name": "ONG KOK E",
            "uuid": "3dcfc622-7b3b-11e8-adc0-fa7ae01bbebc"
          },
          {
            "name": "FAY N",
            "uuid": "26a8c4bc-7b3b-11e8-adc0-fa7ae01bbebc"
          }
        ],
        "teachers": [
          {
            "name": "Amy Choo",
            "uuid": "a2915698-7b3b-11e8-adc0-fa7ae01bbebc"
          },
          {
            "name": "Audrey Tan",
            "uuid": "a924a2bc-7b3b-11e8-adc0-fa7ae01bbebc"
          }
        ]
      }
    }
  }
```


Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Group does not exist or App is not installed to this group.	{ "data":{ "group":null }, "errors":[{ "message":"Exception while fetching data (/group) : Group does not exist" }] }

Retrieve Assignment

Retrieve assignment information via assignment ID.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
{
  assignment(uuid:"d57aa443-43a6-4c58-a692-53865ecd1bca"){
    uuid
    title
    start
    end
    createdBy {
      uuid
      name
    }
    modifiedBy{
      uuid
      name
    }
    tasks(first:2) {
      uuid
      title
      status
    }
  }
}
```

Sample Response

```
 {
  "data": {
    "assignment": {
      "uuid": "d57aa443-43a6-4c58-a692-53865ecd1bca",
      "title": "app lesson1",
      "start": "2018-12-03T17:15:30+08:00",
      "end": "2018-12-13T17:15:30+08:00",
      "createdBy": {
        "uuid": "be15a3ec-7b3b-11e8-adc0-fa7ae01bbebc",
        "name": "Cheng Xin"
      },
      "modifiedBy": null,
      "tasks": [
        {
          "uuid": "b69e0c37-056d-4b9f-aafb-a98e1c7d9dbe",
          "title": "app lesson1",
          "status": "NEW"
        }
      ]
    }
  }
}
```

Error Reponses

HTTP Code	Description	Response

401	Authorization header is invalid.	{
		"errors":[
		{
		"message":"Authorization header is invalid"
		}
		]
		}

200	Assignment does not exist or Assignment is not created from this App.	{
		"data":{
		"assignment":null
		},
		"errors":[
		{
		"message":"Exception while fetching data (
		/assignment) : Assignment does not exist"
		}
		]
		}

Retrieve Task

Retrieve task information via task ID.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
{
  task(uuid: "b69e0c37-056d-4b9f-aafb-a98e1c7d9dbe") {
    uuid
    title
    start
    end
    createdBy {
      name
      uuid
    }
    assignee {
      uuid
      level
      name
      classSerialNo
    }
    subject
    status
  }
}
```

Sample Response


```
{
 "data": {
   "task": {
     "uuid": "b69e0c37-056d-4b9f-aafb-a98e1c7d9dbe",
     "title": "My First Lesson",
     "start": "2018-12-03T17:15:30+08:00",
     "end": "2018-12-13T17:15:30+08:00",
     "createdBy": {
       "name": "Cheng Xin",
       "uuid": "be15a3ec-7b3b-11e8-adc0-fa7ae01bbebc"
     },
     "assignee": {
       "uuid": "d938c6c2-7b3b-11e8-adc0-fa7ae01bbebc",
       "level": "PRIMARY_2",
       "name": "WANG Y",
       "classSerialNo": 10
     },
     "subject": "SCIENCE S1-A",
     "status": "NEW"
   }
  }
}
```


Error Reponses

HTTP Code	Description	Response

401	Authorization header is invalid.	{
		"errors":[
		{
		"message":"Authorization header is invalid"
		}
		]
		}

200	Task does not exist or Task is not created from this App.	{
		"data":{
		"task":null
		},
		"errors":[
		{
		"message":"Exception while fetching data (/task) : Task does not exist"
		}
		]
		}

Retrieve All Groups

Retrieve all groups installed to the app.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
{
  allGroups(changedSince:"2018-12-03T17:15:30+08:00", schoolCode:"
1101", first:50, after:"
Y29tLnVmaW5pdHkuc2xzLmFkZi5tb2RlbC5ncmFwaHFsLkdyb3VwTW9kZWw6Ng") {
    totalCount
    edges {
      node {
        name
        uuid
      }
      cursor
    }
    pageInfo {
      endCursor
      hasNextPage
    }
  }
}
```

Sample Response

```
 {
  "data": {
    "allGroups": {
        "totalCount": 2,
        "edges": [
          {
            "node": {
              "name": "P4-MATHS-1 MATHEMATICS",
                          "uuid": "b69e0c37-056d-4b9f-aafb-a98e1c7d9dbe"
            },
            "cursor":
"Y29tLnVmaW5pdHkuc2xzLmFkZi5tb2RlbC5ncmFwaHFsLkdyb3VwTW9kZWw6MQ"
          },
          {
            "node": {
              "name": "P4-SCI-1 SCIENCE",
                          "uuid": "c92c05c8-cf8d-4f7d-b162-491940ccb87e"
            },
            "cursor":
"Y29tLnVmaW5pdHkuc2xzLmFkZi5tb2RlbC5ncmFwaHFsLkdyb3VwTW9kZWw6Ng"
          }
        ],
        "pageInfo": {
```

```
            "endCursor":
"Y29tLnVmaW5pdHkuc2xzLmFkZi5tb2RlbC5ncmFwaHFsLkdyb3VwTW9kZWw6Ng",
            "hasNextPage": true
          }
        }
    }
}
```

Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Timestamp is invalid.	{ "data":{ "task":null }, "errors":[{ "message":"Exception while fetching data (/allGroups) : Invalid timestamp." }] }
200	School code is invalid.	{ "data":{ "task":null }, "errors":[{ "message":"Exception while fetching data (/allGroups) : School code is invalid." }] }

Create Assignment

Create an assignment when a teacher assigns a resource/lesson/quiz/assignment to students from one group.

- For each assignment, one or more students from the group can be allocated with this assignment, which can be declared through the assignees parameter.
- For each assignee, a Task will automatically be created in SLS corresponding to the given assignment and asignee.
- Changing of the assignees (adding or deleting) can be done through the updateAssignment mutation.

If a resource been assigned to multiple Groups, the app must make multiple createAssignment mutations (for each group) to SLS.

Click here to expand...

URL

```
POST /apis/v1/graphql
```

Sample Request

```
mutation {
  createAssignment(input: {
    title: "My Lesson"
    start:"2017-12-03T10:15:30+01:00"
    end: "2017-12-13T10:15:30+01:00"
    createdBy: "be15a3ec-7b3b-11e8-adc0-fa7ae01bbebc"
    type:LESSON
    assignees:["d938c6c2-7b3b-11e8-adc0-fa7ae01bbebc"]
    groupUuid: "90c639d2-92e1-11e7-b105-0800279a327c"
  }) {
    uuid
  }
}
```

Sample Response

```
 {
  "data": {
    "createAssignment": {
      "uuid": "9b1ddb71-c7e9-4f98-8531-6de5cd0b19a9"
    }
  }
}
```

Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Assignment end date is earlier than start date.	{ "data":{ "createAssignment":null }, "errors":[{ "message":"Exception while fetching data (/createAssignment) : Start is after or equal End" }] }
200	Assignment title is not provided.	{ "data":{ "createAssignment":null }, "errors":[

		{
		"message":"Exception while fetching data (
		/createAssignment) : Field 'Title' is required"
		}
		]
		}

200	Assignee is empty.	{
		"data":{
		"createAssignment":null
		},
		"errors":[
		{
		"message":"Exception while fetching data (
		/createAssignment) : Assignee cannot be empty"
		}
		]
		}

200	Group is not provided.	{
		"data":{
		"createAssignment":null
		},
		"errors":[
		{
		"message":"Exception while fetching data (
		/createAssignment) : Field 'Group' is required"
		}
		]
		}

Update Assignment

Update an assignment - if fields are unchanged, leave them as null.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
mutation {
  updateAssignment(uuid:"d57aa443-43a6-4c58-a692-53865ecd1bca", input: {
    title: "My Newer Lesson"
    start:"2017-12-03T10:15:30+01:00"
    end: "2017-12-05T10:15:30+01:00"
    modifiedBy: "be15a3ec-7b3b-11e8-adc0-fa7ae01bbebc"
    assignees:["3dcfc622-7b3b-11e8-adc0-fa7ae01bbebc","26a8c4bc-7b3b-
11e8-adc0-fa7ae01bbebc"]
  }) {
    uuid
    title
    start
    end
  }
}
```

Sample Response

```
{
 "data": {
   "updateAssignment": {
     "uuid": "d57aa443-43a6-4c58-a692-53865ecd1bca",
     "title": "My Newer Lesson",
     "start": "2018-12-03T17:15:30+08:00",
     "end": "2019-12-05T17:15:30+08:00"
   }
 }
}
```

Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Assignment does no exist or Assignment is not created from this App.	{ "data":{ "updateAssignment":null }, "errors":[{ "message":"Exception while fetching data (/updateAssignment) : Assignment does not exist" }] }
200	Unable to update openInNewTab field.	{ "data":{ "updateAssignment":null }, "errors":[{ "message":"Exception performing the action (/updateAssignment) : Assignment has started, unable to update openInNewTab." }] }

Delete Assignment

Delete an assignment. Return the assignment UUID on success. App can only delete assignments that it has created.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
mutation{
  deleteAssignment(uuid:"b1d590ff-92b2-438f-a2d2-ec2411ea5dab")
}
```

Sample Response

```
{
 "data": {
   "deleteAssignment": "9b1ddb71-c7e9-4f98-8531-6de5cd0b19a9"
 }
}
```

Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Assignment does no exist or Assignment is not created from this App.	{ "data":{ "deleteAssignment":null }, "errors":[{ "message":"Exception while fetching data (/deleteAssignment) : Assignment does not exist" }] }

Update Task

Update the task status to reflect the task progress by the student.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
mutation {
  updateTask(uuid: "ba057b5b-d0e5-43f9-971a-c22c3d27a2d7", status:
IN_PROGRESS){
        status
  }
}
```

Sample Response

```
{
  "data": {
    "updateTask": {
      "status": "IN_PROGRESS"
    }
  }
}
```

Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Task does no exist or Task is not created from this App.	{ "data":{ "updateTask":null }, "errors":[{ "message":"Exception while fetching data (/updateTask) : Task does not exist" }] }

Create Notification

This API allows the creation of a SLS notification, which is a plain text message aimed at one or more recipients of a group.

- If eventType and eventTypeId is provided, recipients who click on this notification can trigger an app launch with the necessary context information.
- If no eventType and eventTypeId is provided, the notification would be a message that is unclickable.

Click here to expand...

URL

`POST /apis/v1/graphql`

Sample Request

```
mutation {
  createNotification(input: {
    message:"Notification"
    scope: GROUP
    scopeId:"90c639d2-92e1-11e7-b105-0800279a327c"
    eventType: "LAUNCH_ASSIGNMENT"
    eventTypeId: "90c639d2-92e1-11e7-b105-0800279a327c"
    recipient:["3dcfc622-7b3b-11e8-adc0-fa7ae01bbebc"]
  }) {
    message
  }
}
```

Sample Response

```
{
 "data": {
   "createNotification": {
     "message": "Notification"
   }
  }
}
```

Error Reponses

HTTP Code	Description	Response
401	Authorization header is invalid.	{ "errors":[{ "message":"Authorization header is invalid" }] }
200	Notification sends to a Group which does not exist or App is not installed to.	{ "data":{ "createNotification":null }, "errors":[{ "message":"Exception while fetching data (/createNotification) : Group does not exist" }] }
200	Notification recipient doesn not exist	{

		"data":{
		"createNotification":null
		},
		"errors":[
		{
		"message":"Exception while fetching data (/createNotification) : Recipient (s) [3dcfc622-7b3b-11e8-adc0-fa7ae01bbebc] does not exist"
		}
		]
		}

Change Log

v1.0.0-20180302

- Initial draft of documentation

v1.1.0-20180629

- Updated User object ID field to UUID

v1.2.0-20180904

- Changed SubjectGroup type to Group to support SLS Group

v1.3.0-20181216

- Added allGroups query to support retrieval of all groups installed with the app
- Added School object in User and Group types to return school information of a user and group respectively
- Updated descriptions and comments

v1.3.1-20190304

- Added openInNewTab field in Assignment type schema to support option to open App within iframe or new tab
- Updated in descriptions and comments

v1.4.1-20200309

- Added new EventType, i.e., LAUNCH_URL for Context to support embedding App in SLS Lesson
- Updated descriptions and comments