Making Requests

- All client requests to the resource server should include the following fields:
    - `identity` the identity used to log in to the auth server
    - `token` the token received back from the auth server
    - `type` the type of request. Should use a ResourceRequestType enum from `enums.py`
    - additional fields as required by the request type
- All Resource server responses will include the following fields
    - `success` a boolean indicating if the requested operation was successful or not
    - `data` a blob of data formatted depending on the request
        - Upon failure, this will simply be a string indicating a reason for the failure.
- Additionally, the resource server will serve all dates as unix epoch ints

Request Types

- `ListLeaderboards`
    - Additional client request fields:
        - None
    - Resource server response `data`:
        - A `list` of leaderboards, each a tuple of the following:
            - id of the leaderboard
            - name of the leaderboard
            - permission of the requesting user (see `enums.py` for values)
- `ShowOneLeaderboard`
    - Additional client request fields:
        - `leaderboard_id`: the id of the leaderboard requested
    - Resource server response `data`:
        - A `dict` containing the following:
            - `id`: the id of the leaderboard
            - `name`: the name of the leaderboard
            - `entries`: a `list` of tuples sorted by score in the proper orientation with the following:
                - the id of the entry
                - the id of the submitting user
                - the identity (name) of the submitting user
                - the submitter's score

- the submission date as an int
- the submission's verified status

- `CreateLeaderboard`
  - Additional client request fields:
    - `leaderboard_name` : the name for the new leaderboard
    - `leaderboard_permission` : the default permission for the new leaderboard
    - `leaderboard_ascending` : `True` to sort scores ascending, `False` to sort descending.
  - Resource server response `data` :
    - The new leaderboard's id

- `AddEntry`
  - Additional client request fields:
    - `leaderboard_id` : id of the leaderboard
    - `score` : the score
    - `comment` : a comment / description
  - Resource server response `data` :
    - the new entry's id

- `ListUsers`
  - Additional client request fields:
    - None
  - Resource server response `data` :
    - a `list` of tuples each with:
      - id of the user
      - identity of the user (username, effectively)

- `ListUnverified`
  - Additional client request fields:
    - `leaderboard_id` : id of the leaderboard
  - Resource server response `data` :
    - a `list` of unverified entries, each a tuple with the following:
      - the entry id
      - the submitting user's id
      - the submitting user's identity
      - entry score
      - date entry submitted

- `GetEntry`
  - Additional client request fields:
    - `entry_id` : the id of the requested entry
  - Resource server response `data` :

- a `dict` with the following fields:
  - `entry`, a tuple of the following items:
    - the id of the entry
    - the id of the submitter
    - the identity of the submitter
    - score of the entry
    - submission date
    - verified state
    - verifying user's id
    - verifying user's identity
    - verification date
  - `comments`: a `list` of tuples, each with the following items:
    - posting user's identity
    - posting date
    - content of the comment
  - `files`: a `list` of tuples, each with the following items:
    - id of the file
    - filename
    - date of submission
- `ViewUser`
  - Additional client request fields:
    - `'user_id`: the id of the requested user
  - Resource server response `data`:
    - a `dict` with the following entries:
      - `user_data`: a tuple of the following items
        - identity of the user
        - registration date
      - `entries`: a `list` of tuples visible to the requesting user, each with the following items:
        - id of the entry
        - id of the leaderboard associated
        - score on the entry
        - verified boolean
        - submission date
- `ViewPermissions`
  - Additional client request fields:
    - `user_id`: the id of the user whose permissions will be viewed

- Resource server response `data` :
    - a `list` of tuples each with:
        - the id of a leaderboard
        - the permission of the user for this leaderboard
- `ModifyEntryVerification`
    - Additional client request fields:
        - `entry_id` : the id of the entry to change
        - `verified` : the state to set (boolean)
    - Resource server response `data` :
        - `None` , use `success` alone to determine outcome.
- `AddComment`
    - Additional client request fields:
        - `entry_id` : the id of the entry to add comment to
        - `content` : the text content of the comment
    - Resource server response `data` :
        - `None` , use `success` alone to determine outcome.
- `RemoveLeaderboard`
    - Removes all comments, entries, and files associated with leaderboard as well as the leaderboard itself
    - Additional client request fields:
        - `leaderboard_id` : the id of the leaderboard to remove
    - Resource server response `data` :
        - `None` , use `success` alone to determine outcome.
- `RemoveEntry`
    - Removes all comments, files and the entry from leaderboard.
    - Additional client request fields:
        - `entry_id` : the id of the entry to remove
    - Resource server response `data`
        - `None` , use `success` alone to determine outcome
- `SetPermission`
    - Sets a user's permission for a leaderboard to a certain permission level. Can update a current permission or set a new permission.
    - Additional client request fields:
        - `user_id` : the id of the user whose permission is to be updated
        - `leaderboard_id` : the leaderboard for the permission
        - `permission` : the new permission for the user
    - Resource server response `data`

  - ■ `None` , use `success` alone to determine outcome
- `RemoveUser`
  - ○ Removes a user and associated data from the database.
  - ○ Additional client request fields:
    - ■ `user_id` : the id of the user to be removed
  - ○ Resource server response `data`
    - ■ `None` , use `success` alone to determine outcome
- `ChangeScoreOrder`
  - ○ Sets a leaderboard's scoring order
  - ○ Additional client request fields:
    - ■ `leaderboard_id` : the id of the leaderboard to be modified
    - ■ `ascending` : boolean corresponding to if the leaderboard should be set to ascending (false for descending)
  - ○ Resource server response `data`
    - ■ `None` , use `success` alone to determine outcome
- `AddProof`
  - ○ adds a proof file associated with an entry. Only the user that submitted the entry can add proof.
  - ○ Additional client request fields:
    - ■ `entry_id` : the id of the entry to associate the file with.
    - ■ `filename` : a filename to associate with the file
    - ■ `file` : a byte blob of the file
  - ○ Resource server response `data`
    - ■ `None` , use `success` alone to determine outcome
- `DownloadProof`
  - ○ Download a proof file from the server
  - ○ Additional client request fields:
    - ■ `file_id` : the id of the file to download
  - ○ Resource server response `data`
    - ■ the requested file as a byte blob
- `GetIdFromIdentity`
  - ○ Get a user's id from their identity
  - ○ Additional client request fields:
    - ■ `identity` : the identity of the desired user
  - ○ Resource server response `data` :
    - ■ the id of the desired user
- `ListAccessGroups`

- List users and their access level for a particular leaderboard. Requires moderator permissions.
- Additional client request fields:
  - `leaderboard_id` : the id of the leaderboard in question
- Resource server response `data` :
  - a `list` of users, each a tuple with the following fields:
    - id of the user
    - identity of the user
    - their access level
- `RemoveProof`
  - Remove a proof file from the server. Can only be done by the submitter or a moderator.
  - Additional client request fields:
    - `file_id` : the id of the file to be deleted.
  - Resource server response `data` :
    - `None` . Use `success` field alone to determine outcome.