All APIs return 2xx for both Success and Fail requests, but return 4xx for fatal errors in the server.

User

1. api/userRegister
    - Though the user registration is via a page of eratos, **be sure to call this after a new user is registered**, to send the newly registered user info to us, otherwise there will be errors in the database
    - Method: post
    - Parameters: None
    - Payload: Json<**User**>
        - Example of Json<User> (the one get from eratos api)


```
{
    "auth0Id": "auth0|605851797905250069 01be39",
    "id": "https://staging.e-pn.io/users/xsrudeulfvtr2eiaiksic2jf",
    "info": "https://staging.e-pn.io/resources/gu4km2nnimvyqepmaw3mfcwg",
    "resourcePolicy": "https://staging.e-pn.io/policies/3ylpwlaj6o3wb7p4ui4kkeie",
    "termsLastAcceptedDate": "2021-03-25T03:13:39.912183Z",
    "termsLastAcceptedVersion": 12
}
```


        - Note: it is just the Json you get by calling "https://staging.e-tr.io/auth/me" using a valid token (of the new user)
    - Response:
        - Success:
            - {"Success": "True", "UserName": "the name"}
        - Fail:
            - {"Success": "False", "Message": "error message"}
2. api/getUserInfo
    - Method: get
    - Parameters:
        - userUri: the uri of a user, or "all" for getting all users
        - (Only needed when userUri=all) start: Specify the start index of the user table
        - (Only needed when userUri=all) end: Specify the end index of the user table
            - Note: end-start must be <= 100. Use a loop to get more than 100 users.
    - Payload: none
    - Response:
        - Success:
            - {"Success": "True", "UserInfo": "String of Json<**UserTable**>"}

- {“Success”: “True”, “UserInfo”: [String of Json<UserTable>s, separated by ‘,’]}
- Structure of Json<UserTable>

```
11 references
public int UserID { get; set; }
5 references
public string EratosUserID { get; set; }
8 references
public string Email { get; set; }
7 references
public string Name { get; set; }
5 references
public string Auth0ID { get; set; }
4 references
public string CreatedAt { get; set; }
9 references
public string Info { get; set; }
9 references
public bool isAdmin { get; set; }
```

- Fail:
  - {“Success”: “False”, “Message”: “error message”}

3. api/updateUserInfo
   - Method: post
   - Parameters: none
   - Payload: Json<**UserTable**> (the one get from calling our api getUserInfo, not the one get from the eratos api.)
   - Response:
     - Success:
       - {“Success”: “True”, “Message”: “The user info is up to date.”}
     - Fail:
       - {“Success”: “False”, “Message”: “error message”}

Module

1. api/createModifyModule
   - (for creating new modules, or modifying the availability of modules)
   - Method: post
   - Parameters
     - moduleSchema: link to Eratos schema, i.e: "https://schemas.eratos.ai/json/person"
     - moduleName: name of the module, i.e: Person, Climate Data, etc
     - isActive: (true or false, in lower cases)
   - Payload: none
   - Response:
     - Success:
       - {“Success”: “True”}
     - Fail:
       - {“Success”: “False”, “Message”: “error message”}
2. api/getAllModules
   - Method: get
   - Parameter:

        - Start: integer number. Index of the module
        - End: integer number. Index of the module
            - Note: end-start must be <= 100. Use a loop to get more than 100 items.
    - Payload: none
    - Response:
        - Success:
            - {“Success”: “True”, “Modules”: [String of Json<Module>s, separated by ‘,’]}
                - Format of Json<Module>

```
5 references
public int ModuleID { get; set; }
5 references
public string ModuleName { get; set; }
6 references
public string ModuleSchema { get; set; }
6 references
public bool isActive { get; set; }
```

        - Fail:
            - {“Success”: “False”, “Message”: “error message”}
3. api/getActiveModules
    - Method: get
    - Parameter: none
    - Payload: none
    - Response:
        - Success:
            - {“Success”: “True”, “Modules”: [String of Json<Module>s, separated by ‘,’]}
            - Format of Json<Module>

```
5 references
public int ModuleID { get; set; }
5 references
public string ModuleName { get; set; }
6 references
public string ModuleSchema { get; set; }
6 references
public bool isActive { get; set; }
```

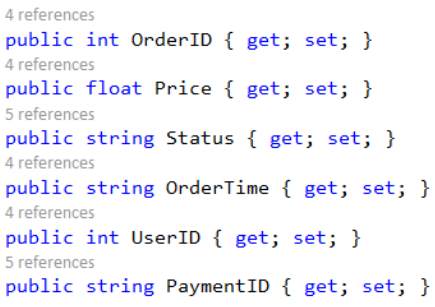
        - Fail:
            - {“Success”: “False”, “Message”: “error message”}

Task & Order

1. api/createTask
    - Method: post
    - Parameters:
        - userUri (e.g. "https://staging.e-pn.io/users/xsrudeulfvtr2eiaiksic2jf")

        - paymentID (any unique string, for example userUri + timestamp etc., you decide)
        - price (must be a string that can be parsed into float)
        - moduleType (the uri of type scheme, e.g. "https://schemas.eratos.ai/json/person")
        - taskType (the type of processing e.g. "GenerateClimateInfo")
        - name (the name of the task & resource, can be any string)
        - geometry (must be in the correct wkt format, e.g. "POINT(140.3142799 -42.84756651)")
        - priority (“low”, “normal” or “high”)
    - Payload: None
    - Response:
        - Success:
            - {“Success”: “True”, “TaskID”: “task id”}
        - Fail:
            - {“Success”: “False”, “Message”: “error message”}
2. api/getTasksOrdersOfUser
        - Get all tasks and orders of a user
    - Method: get
    - Parameters:
        - userUri (e.g. "https://staging.e-pn.io/users/xsrudeulfvtr2eiaiksic2jf")
    - Payload: None
    - Response:
        - Success:
            - {“Success”: “True”, “Tasks”: [String of Json<Task>s, separated by ‘,’], “Orders”: [String of Json<Order>s, separated by ‘,’]}
            - Format of Json<Task>

```
10 references
public int TaskID { get; set; }
9 references
public string EratosTaskID { get; set; }
4 references
public string CreatedAt { get; set; }
5 references
public string LastUpdatedAt { get; set; }
5 references
public string StartedAt { get; set; }
5 references
public string EndedAt { get; set; }
5 references
public string Name { get; set; }
5 references
public string Priority { get; set; }
9 references
public string State { get; set; }
5 references
public string Type { get; set; }
5 references
public string Meta { get; set; }
5 references
public string Error { get; set; }
4 references
public int UserID { get; set; }
5 references
public int OrderID { get; set; }
```

        - Note: “Meta” is the uri of the resource, which is used for viewing the final processed result. Click the uri after the state becomes “Complete”, the user should be able to view the result. The resource uri will always be there but doesn’t work before the task is completed.
    - Format of Json<Order>
        - 

    - Fail:
        - {“Success”: “False”, “Message”: “error message.”}
3. api/syncTasksAndOrders
    - This function is to sync our database with the eratos server. Our database will automatically run this after a specified time, but if you want to get the newest information of tasks and orders, you can call this manually before you get task and order information.
    - Method: get
    - Parameters: none
    - Payload: none
    - Response:
        - Success:
            - {“Success”: “True”, “Message”: “Database is up to date.”}
        - Fail:
            - {“Success”: “False”, “Message”: “Sync error.”}