Client routes

These routes relate to clients and their attributes

**Get all clients**

|  |  |
| --- | --- |
| Route | /client/ |
| Method | GET |
| Input | none |
| Output | Array of Client objects |
| Description | This route gets a list of all of the clients. We may in the future deprecate this for a paginated version, but for now we will load all client data in one shot. |

**Get a client**

|  |  |
| --- | --- |
| Route | /client/:id |
| Method | GET |
| Input | none |
| Output | A specific client or an error |
| Description | This route gets a specific clients by id. |

**Create a new client**

|  |  |
| --- | --- |
| Route | /client/new |
| Method | POST |
| Input | Array of ClientAttribute objects |
| Output | The client object or an error |
| Description | Creates a new client and returns the new client |

**Update a client**

|  |  |
| --- | --- |
| Route | /client/:id/update |
| Method | POST |
| Input | A Client object |
| Output | The updated client object or an error |
| Description | Updates a client and returns the updated client. The attributes belonging to the input client object must be the full list of attributes belonging to the client, even if only one of the attributes has changed. |

**Delete a client**

|  |  |
| --- | --- |
| Route | /client/:id/delete |
| Method | POST |
| Input | None |
| Output | Success or an error |
| Description | Deletes a client |

Client Attribute Type routes

These routes relate to client attribute types

**Get a list of all ClientAttributeTypes**

|  |  |
| --- | --- |
| Route | /clientAttributeType/ |
| Method | GET |
| Input | none |
| Output | Array[Object ClientAttributeType], sorted by its ordering field |
| Description | Returns a list of ClientAttributeTypes. This is useful for generating forms where users can input or update client information. These are also used for determining if onboarding is complete. |

**Create a ClientAttributeTypes**

|  |  |
| --- | --- |
| Route | /clientAttributeType/new |
| Method | POST |
| Input | Array[Object ClientAttributeType] |
| Output | Array[Object ClientAttributeType] |
| Description | Upserts the list of ClientAttributeTypes. If a record exists in the database and is not present in this list, it is not touched. |

**Update a ClientAttributeTypes**

|  |  |
| --- | --- |
| Route | /clientAttributeType/:attribName |
| Method | POST |
| Input | Object ClientAttributeType |
| Output | Object ClientAttributeType |
| Description | Update the ClientAttributeTypes. |

**Delete ClientAttributeTypes**

|  |  |
| --- | --- |
| Route | /clientAttributeType/:attribName/delete |
| Method | POST |
| Input | None |
| Output | Success or error |
| Description | Deletes the ClientAttributeTypes whose attribName is supplied in the input to this endpoint. |

Object definitions:

Object Client

{

“id”: String = UUID of user

“attributes”: { internal\_attrib\_name => object ClientAttribute} = client attributes

}

Object ClientAttribute

{

“type”: ClientAttributeType = type of the attribute

“value”: Any = string representation of the attribute. The “attribDataType” of the ClientAttributeType will determine how this is displayed on screen and changed. It can

}

Object ClientAttributeType

{

“attribName”: String = name of attribute. This is used for internal reference, and looks something like “date\_of\_birth”.

“attribDisplayName”: String = display name of attribute. This is used for display purposes, and looks like “Date of birth”

“attribDataType”: String = data type of attribute. Used for display purposes. I will shortly generate a list of acceptable attribDataTypes, but for now it is (“string”, “date”, “boolean”)

“attribRequired”: Boolean = record required to save client information

“attribRequiredForOnboarding”: Boolean = record required to claim that the client has been onboarded

“ordering”: Int = often times, attributes will want to be displayed in some sort of order. This field controls that

}

Object Error

{

“status”: “error”

“error”: String = class name of the Java exception thrown

“message”: String = a string with details about the error

}

Object Success

{

“status”: “success”

“message”: String = message giving details.

}