MeshyDB Documentation

Yetisoftworks

Jan 24, 2020

Getting Started

CHAPTER 1

What is Meshy?

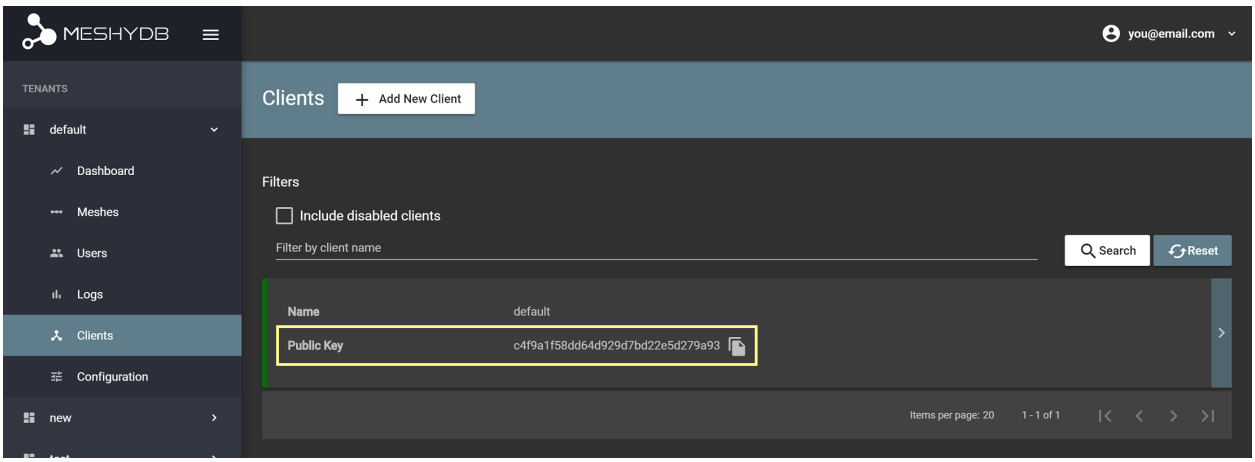
MeshyDB gives you a fully functional API backend in minutes. We take care of the bulky time-consuming API, letting you focus on the front-end design. Build apps faster by leveraging the MeshyDB backend.

CHAPTER 2

Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once your account is verified, you will need to gather your from the Clients page under your default tenant. See image below:



CHAPTER 3

Next Steps

Now that you have your public key, you can begin with any of our language specific quick starts:

- C#
- NodeJS
- REST

3.1 C#

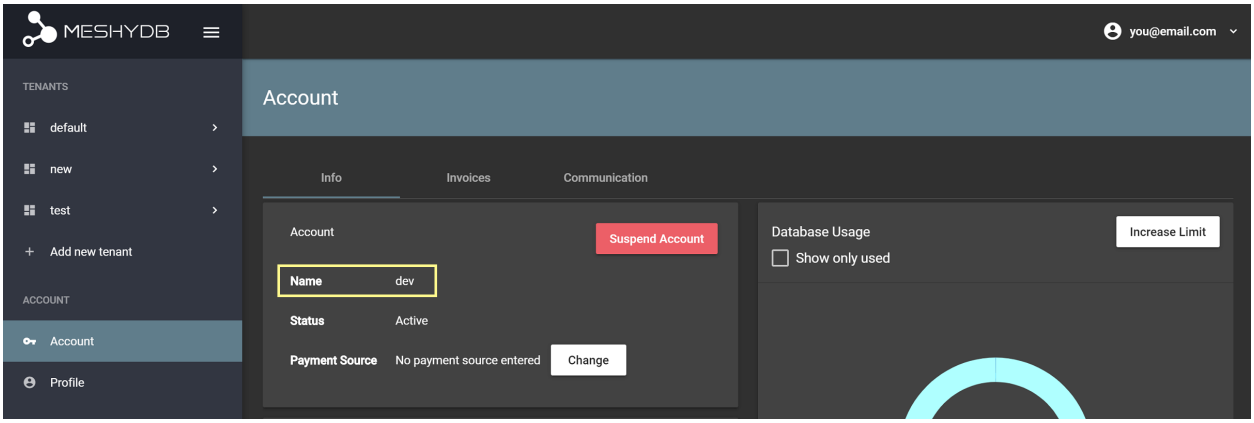
3.1.1 Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once you verify your account you will need to gather your and .

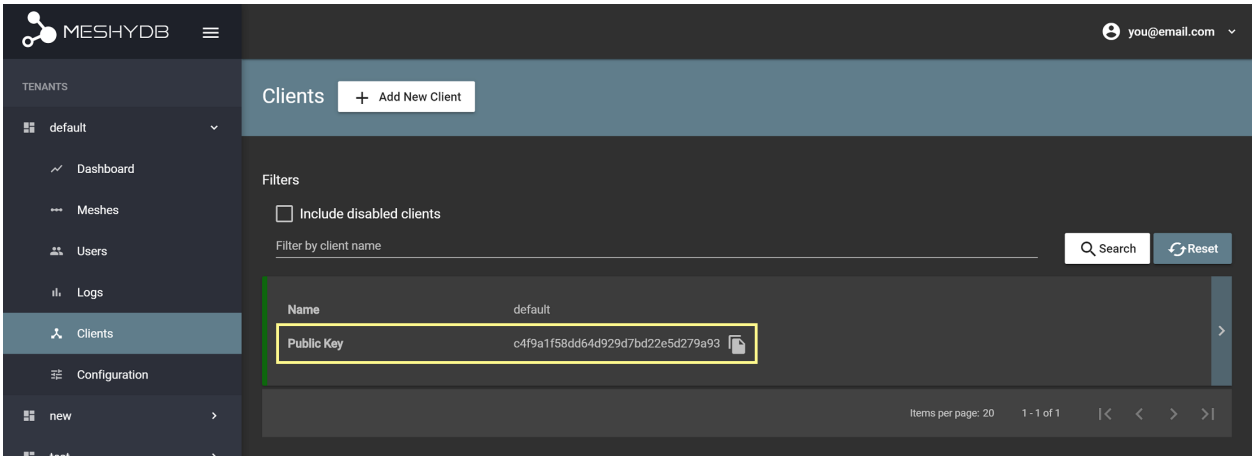
Identify Account Name

Your can be found under the Account page. See image below:



Identify Public Key

Your can be found under the Clients page under your default tenant. See image below:



In the following we will assume no other configuration has been made to your account or tenants so we can just begin!

3.1.2 Install SDK

The supporting SDK is open source and you are able to use .Net Framework 4.7.1+ or .Net Core 2.x.

Let's install the MeshyDB.SDK NuGet package with the following command:

```
Install-Package MeshyDb.SDK
```

3.1.3 Initialize

The client is used to establish a connection to the API. You will need to provide your and from your account and client.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

3.1.4 Register Anonymous User

Anonymous users are great for associating data to people or devices without having them go through any type of user registration.

The example below shows verifying a username is available and registering an anonymous user if the username does not exist.

C#

```
var username = "mctesterton";

var userExists = await client.CheckUserExistAsync(username);

if (!userExists.Exists)
{
   await client.RegisterAnonymousUserAsync(username);
}
```

username [string] Unique user name for authentication. If it is not provided a username will be automatically generated.

Responses

201 [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

400 [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.1.5 Login

All data interaction must be done on behalf of a user. This is done to ensure proper authorized access of your data.

The example below shows logging in an anonymous user.

C#

```
var connection = await client.LoginAnonymouslyAsync(username);
```

username [string, required] Unique user name for authentication.

Responses

200 [OK]

- Generates new credentials for authorized user.

Example Result

```
{
  "access_token": "ey...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "ab23cd3343e9328g"
}
```

400 [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

Once we login we can access our connection through a static member.

C#

```
connection = MeshyClient.CurrentConnection;
```

3.1.6 Retrieving Self

When a user is created they have some profile information that helps identify them. We can use this information to link their id back to data that has been created.

The example below shows retrieving information of the user.

C#

```
var user = await connection.Users.GetSelfAsync();
```

No parameters provided.

Responses

200 [OK]

- Retrieves information about the authorized user.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.1.7 Create data

Now that a user connection is established you can begin making API requests.

The MeshyDB SDK requires all data extend the class.

The example below shows a Person represented by a first name, last name and user id.

C#

```
// Mesh Name can be overridden by attribute, otherwise by default it is derived from␣
↪class name
[MeshName("Person")]
public class Person : MeshData
{
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public string UserId { get; set; }
}
```

Now that we have a representation of a person we can start making data to write to the API.

The example below shows committing a new person.

C#

```
var model = new Person()
{
   FirstName = "Bob",
   LastName = "Bobson",
   UserId = user.Id
};

model = await connection.Meshes.CreateAsync(model);
```

model [object, required] Representation of data that *must* extend .

Responses

201 [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Bob",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.1.8 Update data

The API allows you to make updates to specific by targeting the id.

The SDK makes this even simpler since the id can be derived from the object itself along with all it's modifications.

The example below shows modifying the first name and committing those changes to the API.

C#

```
model.FirstName = "Robert";

model = await connection.Meshes.UpdateAsync(model);
```

model [object, required] Representation of data that *must* extend .

Responses

200 [OK]

- Result of updated mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Robert",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.1.9 Search data

The API allows you to search using a Linq expression.

The example below shows searching based where the first name starts with Rob.

C#

```
Expression<Func<Person, bool>> filter = (Person x) => x.FirstName.StartsWith("Rob");

var pagedPersonResult = await connection.Meshes
                                        .SearchAsync<Person>(filter);
```

filter [object] Criteria provided in a Linq expression to limit results.

Responses

200 [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results":  [{
```

(continues on next page)

(continued from previous page)

```
                    "_id":"5d438ff23b0b7dd957a765ce",
                    "firstName": "Robert",
                    "lastName": "Bobson",
                    "userId": "5c78cc81dd870827a8e7b6c4"
                }],
    "totalRecords": 1
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

In some cases you may need more control on your filtering or ordering. You can optionally provide this criteria in a MongoDB format.

3.1.10 Delete data

The API allows you to delete a specific by targeting the id.

The example below shows deleting the data from the API by providing the object.

Deleted data is not able to be recovered. If you anticipate the need to recover this data please implementing a .

C#

```
var id = model.Id;

await connection.Meshes.DeleteAsync<Person>(id);
```

id [string, required] Identifier of record that must be deleted.

Responses

204 [No Content]

- Mesh has been deleted successfully.

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete meshes or mesh.

404 [Not Found]

- Mesh data was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.1.11 Sign out

The MeshyDB SDK manages a single connection to the API.

The Meshy SDK handles token management, this includes refresh tokens used to maintain a user's connection.

As a result it is recommended to implement Sign Out to ensure the current user is logged out and all refresh tokens are revoked.

The example below shows signing out of the currently established connection.

C#

```
await connection.SignoutAsync();
```

No parameters provided. The connection is aware of who needs to be signed out.

Responses

200 [OK]

- Identifies successful logout.

400 [Bad request]

- Invalid client id.
- Token is missing.
- Unsupported Token type.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

Not seeing something you need? Feel free to give us a chat or contact us at support@meshydb.com.

3.2 NodeJS

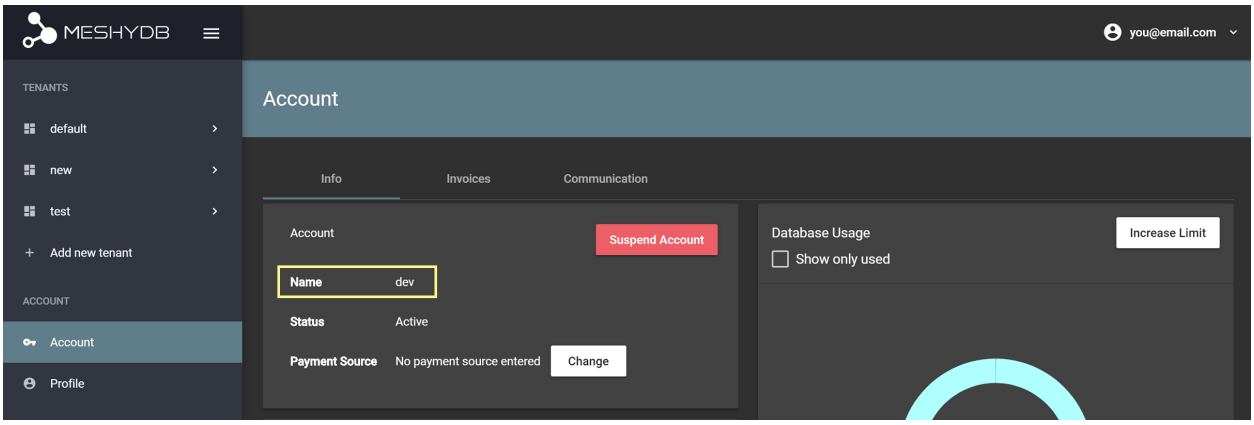
3.2.1 Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once you verify your account you will need to gather your and .

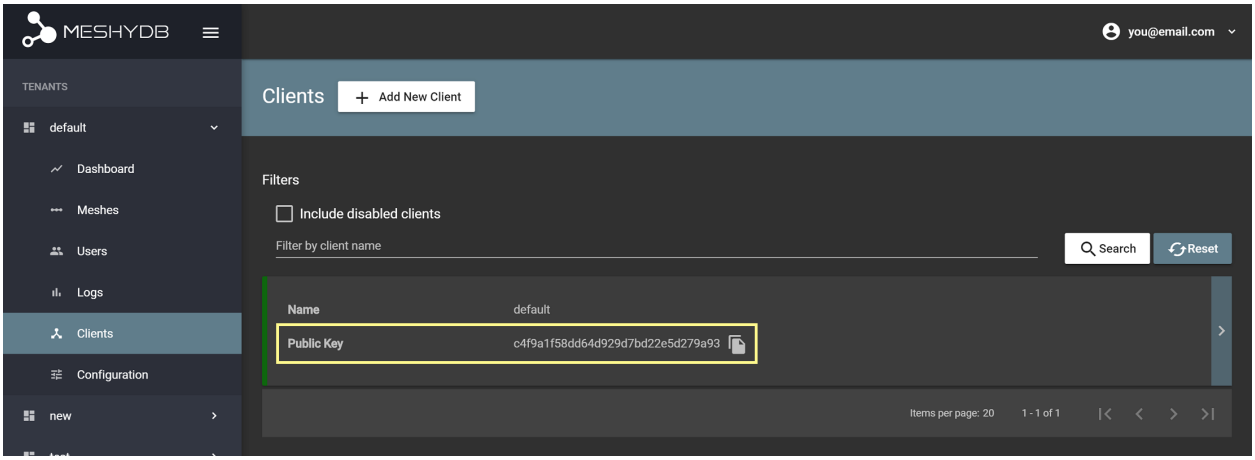
Identify Account Name

Your can be found under the Account page. See image below:



Identify Public Key

Your can be found under the Clients page under your default tenant. See image below:



In the following we will assume no other configuration has been made to your account or tenants so we can just begin!

3.2.2 Install SDK

The supporting SDK is open source and you are able to use a browser or NodeJS application.

Let's install the MeshyDB.SDK NPM package with the following command:

```
npm install @meshydb/sdk
```

3.2.3 Initialize

The client is used to establish a connection to the API. You will need to provide your and from your account and client.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

3.2.4 Register Anonymous User

Anonymous users are great for associating data to people or devices without having them go through any type of user registration.

The example below shows verifying a username is available and registering an anonymous user if the username does not exist.

NodeJS

```
var username = "mctesterton";

var userExists = await client.checkUserExist(username);

if (!userExists.exists) {
   await client.registerAnonymousUser(username);
}
```

username [string] Unique user name for authentication. If it is not provided a username will be automatically generated.

Responses

201 [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

400 [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.2.5 Login

All data interaction must be done on behalf of a user. This is done to ensure proper authorized access of your data.

The example below shows logging in an anonymous user.

NodeJS

```
var connection = await client.loginAnonymously(username);
```

username [string, required] Unique user name for authentication.

Responses

200 [OK]

- Generates new credentials for authorized user.

Example Result

```
{
  "access_token": "ey...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "ab23cd3343e9328g"
}
```

400 [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

Once we login we can access our connection through a static member.

NodeJS

```
connection = MeshyClient.currentConnection;
```

3.2.6 Retrieving Self

When a user is created they have some profile information that helps identify them. We can use this information to link their id back to data that has been created.

The example below shows retrieving information of the user.

NodeJS

```
var user = await connection.usersService.getSelf();
```

No parameters provided.

Responses

200 [OK]

- Retrieves information about the authorized user.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.2.7 Create data

Now that a user connection is established you can begin making API requests.

The example below shows committing a new person.

NodeJS

```
var model = {
                _id: undefined,
                firstName: "Bob",
                lastName: "Bobson",
                userId: user.id
            };

var meshName = "person";

model = await connection.meshesService.create(meshName, model);
```

meshName [string, required] Identifies which mesh collection to manage.

model [object, required] Represents a person in this example.

Responses

201 [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Bob",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.2.8 Update data

The API allows you to make updates to specific Mesh Data by targeting the id.

The SDK makes this even simpler since the id can be derived from the object itself along with all it's modifications.

The example below shows modifying the first name and committing those changes to the API.

NodeJS

```
model.firstName = "Robert";

model = await connection.meshesService.update(meshName, model);
```

meshName [string, required] Identifies which mesh collection to manage.

model [object, required] Represents a person in this example.

Responses

200 [OK]

- Result of updated mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Robert",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.2.9 Search data

The API allows you to search a mesh collection using a MongoDB expression.

The example below shows searching based where the first name starts with Rob.

NodeJS

```
var filter = { 'firstName': { "$regex": "^Rob" } };

var pagedPersonResult = await connection.meshesService
                                        .search(meshName, { filter: filter });
```

meshName [string, required] Identifies which mesh collection to manage.

filter [object] Criteria provided in a MongoDB expression to limit results.

Responses

200 [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results":  [{
                  "_id":"5d438ff23b0b7dd957a765ce",
                  "firstName": "Robert",
                  "lastName": "Bobson",
                  "userId": "5c78cc81dd870827a8e7b6c4"
               }],
```

(continues on next page)

(continued from previous page)

```
    "totalRecords": 1
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.2.10 Delete data

The API allows you to delete a specific Mesh Data by targeting the id.

The example below shows deleting the data from the API by providing the object.

Deleted data is not able to be recovered. If you anticipate the need to recover this data please implementing a .

NodeJS

```
var id = model._id;

await connection.meshesService.delete(meshName, id);
```

meshName [string, required] Identifies which mesh collection to manage.

id [string, required] Identifier of record that must be deleted.

Responses

204 [No Content]

- Mesh has been deleted successfully.

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete meshes or mesh.

404 [Not Found]

- Mesh data was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.2.11 Sign out

The MeshyDB SDK manages a single connection to the API.

The Meshy SDK handles token management, this includes refresh tokens used to maintain a user's connection.

As a result it is recommended to implement Sign Out to ensure the current user is logged out and all refresh tokens are revoked.

The example below shows signing out of the currently established connection.

NodeJS

```
await connection.signout();
```

No parameters provided. The connection is aware of who needs to be signed out.

Responses

200 [OK]

- Identifies successful logout.

400 [Bad request]

- Invalid client id.
- Token is missing.
- Unsupported Token type.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

Not seeing something you need? Feel free to give us a chat or contact us at support@meshydb.com.

3.3 REST

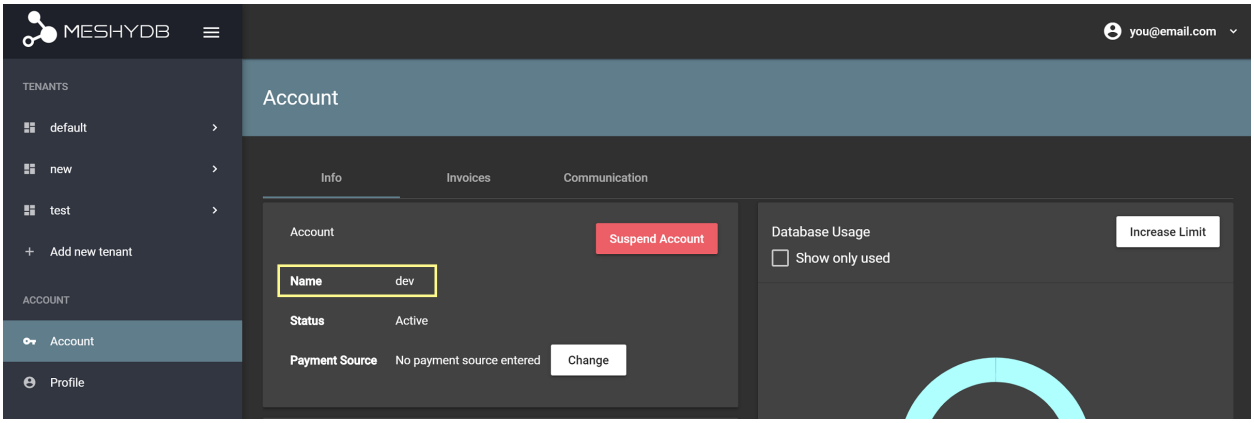
3.3.1 Before you get started!

This documentation assumes you have an active MeshyDB account. If you do not, please create a free account at https://meshydb.com.

Once you verify your account you will need to gather your and .

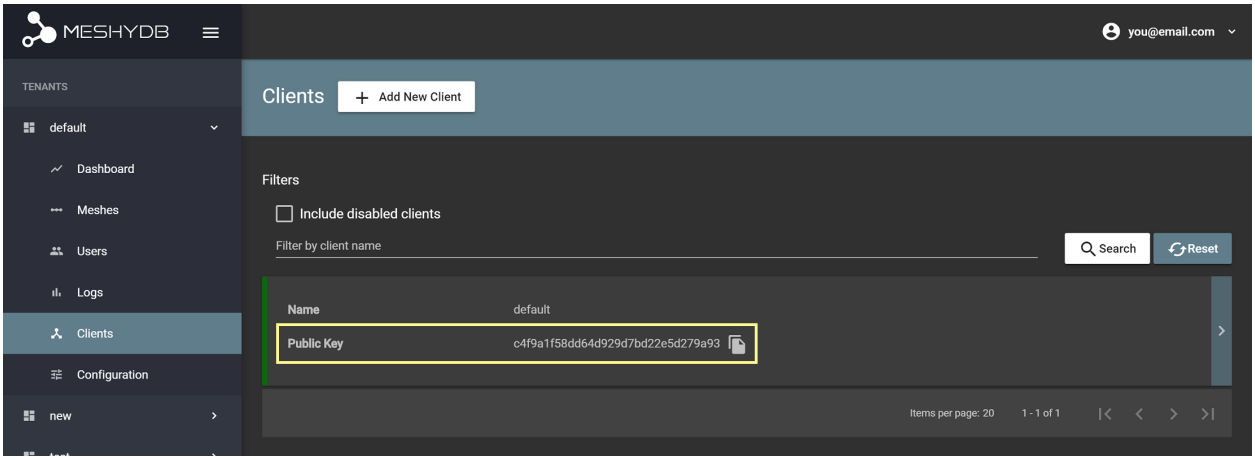
Identify Account Name

Your can be found under the Account page. See image below:



Identify Public Key

Your can be found under the Clients page under your default tenant. See image below:



In the following we will assume no other configuration has been made to your account or tenants so we can just begin!

3.3.2 Checking Username Exists

Checking username helps identify if a device or user has already registered.

The example below shows verifying a username is available.

REST

```
GET https://api.meshydb.com/{accountName}/users/{username}/exists HTTP/1.1
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

Responses

201 [Created]

- Identifies if username already exists.

Example Result

```
{
   "exists": false
}
```

400 [Bad request]

- Username is required.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.3 Register Anonymous User

Anonymous users are great for associating data to people or devices without having them go through any type of user registration.

The example below shows registering an anonymous user.

REST

```
POST https://api.meshydb.com/{accountName}/users/register/anonymous HTTP/1.1
Content-Type: application/json

  {
    "username": "mctesterton"
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

Responses

201 [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

400 [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.4 Login

All data interaction must be done on behalf of a user. This is done to ensure proper authorized access of your data.

The example below shows logging in an anonymous user.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

   client_id={publicKey}&
   grant_type=password&
   username={username}&
   password=nopassword&
   scope=meshy.api offline_access
```

(Form-encoding removed, and line breaks added for readability)

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

Responses

200 [OK]

- Generates new credentials for authorized user. The token will expire and will need to be refreshed.

Example Result

```
{
  "access_token": "ey...",
  "expires_in": 3600,
  "token_type": "Bearer",
  "refresh_token": "ab23cd3343e9328g"
}
```

400 [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.

- Invalid Scope.
- Username is invalid.
- Password is invalid.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.5 Retrieving Self

When a user is created they have some profile information that helps identify them. We can use this information to link their id back to data that has been created.

The example below shows retrieving information of the user.

REST

```
GET https://api.meshydb.com/{accountName}/users/me HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

Responses

200 [OK]

- Retrieves information about the authorized user.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "mctesterton",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.6 Create data

Now that a user is authorized you can begin making API requests.

The example below shows committing a new such as a person.

REST

```
POST https://api.meshydb.com/{accountName}/meshes/{meshName} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

   {
      "firstName": "Bob",
      "lastName": "Bobson",
      "userId": "5c78cc81dd870827a8e7b6c4"
   }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during *Login*.

meshName [string, required] Identifies name of mesh collection. e.g. person.

Responses

201 [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Bob",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.7 Update data

The API allows you to make updates to specific by targeting the id.

The example below shows modifying the first name and committing those changes to the API.

REST

```
PUT https://api.meshydb.com/{accountName}/meshes/{meshName}/{id}  HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

   {
      "firstName": "Robert",
      "lastName": "Bobson"
   }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during *Login*.

meshName [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to replace.

Responses

200 [OK]

- Result of updated mesh data.

Example Result

```
{
   "_id":"5d438ff23b0b7dd957a765ce",
   "firstName": "Robert",
   "lastName": "Bobson",
   "userId": "5c78cc81dd870827a8e7b6c4"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.8 Search data

The API allows you to search using a MongoDB expression.

The example below shows searching based where the first name starts with Rob.

REST

```
GET https://api.meshydb.com/{accountName}/meshes/{meshName}?filter={ 'firstName': { "
↪$regex": "^Rob" } } HTTP/1.1
Authorization: Bearer {access_token}
```

(Encoding removed for readability)

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during *Login*.

meshName [string, required] Identifies name of mesh collection. e.g. person.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [string] Defines which fields need to be ordering and direction in a MongoDB format.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results":  [{
                  "_id":"5d438ff23b0b7dd957a765ce",
                  "firstName": "Robert",
                  "lastName": "Bobson",
                  "userId": "5c78cc81dd870827a8e7b6c4"
               }],
   "totalRecords": 1
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.9 Delete data

The API allows you to delete a specific by targeting the id.

The example below shows deleting the data from the API by providing the object.

Deleted data is not able to be recovered. If you anticipate the need to recover this data please implementing a .

REST

```
DELETE https://api.meshydb.com/{accountName}/meshes/{meshName}/{id} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during *Login*.

meshName [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to remove.

Responses

204 [No Content]

- Mesh has been deleted successfully.

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete meshes or mesh.

404 [Not Found]

- Mesh data was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.3.10 Sign out

When a user is authenticated a refresh token is generated. The refresh token allows a user to be silently authenticated.

As a result it is recommended to implement Sign Out to ensure the current user is logged out and all refresh tokens are revoked.

The example below shows revoking the refresh token. The access token is short lived and will expire within an hour.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/revocation HTTP/1.1
Content-Type: application/x-www-form-urlencoded

  client_id={accountName}&
```

(continues on next page)

(continued from previous page)

```
  grant_type=refresh_token&
  token={refresh_token}
```

(Line breaks added for readability)

accountName [string, required] Indicates which account you are connecting to.

refresh_token [string, required] Token to allow reauthorization with MeshyDB after the access token expires requested during *Login*.

Responses

200 [OK]

- Identifies successful logout.

400 [Bad request]

- Invalid client id.
- Token is missing.
- Unsupported Token type.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

Not seeing something you need? Feel free to give us a chat or contact us at support@meshydb.com.

3.4 Generating Token

Create a short lived access token to be used for authorized API calls. Typically a token will last 3600 seconds(one hour).

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = client.LoginWithPassword(username, password);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username,password);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

   client_id={publicKey}&
   grant_type=password&
   username={username}&
   password={password}&
   scope=meshy.api offline_access
```

(Form-encoding removed, and line breaks added for readability)

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

Responses

200 [OK]

- Generates new credentials for authorized user.

Example Result

```
{
   "access_token": "ey...",
   "expires_in": 3600,
   "token_type": "Bearer",
   "refresh_token": "ab23cd3343e9328g"
}
```

400 [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Invalid Scope.
- Username is invalid.
- Password is invalid.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.5 Refreshing Token

Using the token request made to generate an access token, a refresh token will also be generated.

Once the token expires the refresh token can be used to generate a new set of credentials for authorized calls.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = client.LoginWithPassword(username, password);
var refreshToken = connection.RetrieveRefreshToken();

connection = await client.LoginWithRefreshAsync(refreshToken);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

refreshToken [string, required] Refresh token generated from previous access token generation.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username,password);

var refreshToken = meshyConnection.retrieveRefreshToken();

var refreshedMeshyConnection = await client.loginWithRefresh(refreshToken);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

refreshToken [string, required] Refresh token generated from previous access token generation.

REST

```
POST https://auth.meshydb.com/{accountName}/connect/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded

   client_id={publicKey}&
   grant_type=refresh_token&
   refresh_token={refresh_token}
```

(Form-encoding removed, and line breaks added for readability)

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

refresh_token [string, required] Refresh token generated from previous access token generation.

Responses

200 [OK]

- Generates new refresh credentials for authorized user.

Example Result

```
{
   "access_token": "ey...",
   "expires_in": 3600,
   "token_type": "Bearer",
   "refresh_token": "ab23cd3343e9328g"
}
```

400 [Bad request]

- Token is invalid.
- Client id is invalid.
- Grant type is invalid.
- User is no longer active.
- Refresh token is expired.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.6 Creating Users

A user can be authenticated with the system for ensuring they are authorized to interact with the system.

You can either generate an anonymous user, or device user with limited functionality. Otherwise you can register a new user with full credentials.

3.6.1 Checking Username Available

Before identifying a device or unique user you can check to see if they already were registered.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var userExists = await client.CheckUserExistAsync(username);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var userExists = await client.checkUserExist(username);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

REST

```
GET https://api.meshydb.com/{accountName}/users/{username}/exists HTTP/1.1
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

Responses

201 [Created]

- Identifies if username already exists.

Example Result

```
{
   "exists": false
}
```

400 [Bad request]

- Username is required.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.6.2 Registering Anonymous User

An anonymous user can identify a device or unique user without requiring user interaction.

This kind of user has limited functionality such as not having the ability to be verified or be assigned roles.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var anonymousUser = await client.RegisterAnonymousUserAsync(userName);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser(username);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

REST

```
POST https://api.meshydb.com/{accountName}/users/register/anonymous HTTP/1.1
Content-Type: application/json

  {
    "username": "username_testermctesterson"
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

Responses

201 [Created]

- New user has been registered and is now available for use.

Example Result

```
{
   "id": "5c78cc81dd870827a8e7b6c4",
   "username": "username_testermctesterson",
   "firstName": null,
   "lastName": null,
   "verified": false,
   "isActive": true,
   "phoneNumber": null,
   "emailAddress": null,
   "roles": [],
   "securityQuestions": [],
   "anonymous": true,
   "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

400 [Bad request]

- Username is a required field.
- Anonymous registration is not enabled.
- Username must be unique.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.6.3 Registering User

Registering a user allows user defined credentials to access the system.

If email or text verification is configured, they will be prompted to verify their account.

The user will not be able to be authenticated until verification has been completed. The verification request lasts one hour before it expires.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var user = new RegisterUser();

await client.RegisterUserAsync(user);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

newPassword [string, required] New user secret credentials for login.

firstName [string] First name of registering user.

lastName [string] Last name of registering user.

phoneNumber [string, required *if using phone verification*] Phone number of registering user.

emailAddress [string, required *if using email verification*] Email address of registering user.

securityQuestions [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var user = await client.registerUser({
                                        username: username,
                                        newPassword: newPassword,
                                        firstName: firstName,
                                        lastName: lastName,
                                        phoneNumber: phoneNumber,
                                        emailAddress: emailAddress,
                                        securityQuestions: securityQuestions
                                    });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

newPassword [string, required] New user secret credentials for login.

firstName [string] First name of registering user.

lastName [string] Last name of registering user.

phoneNumber [string, required *if using phone verification*] Phone number of registering user.

emailAddress [string, required *if using email verification*] Email address of registering user.

securityQuestions [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

REST

```
POST https://api.meshydb.com/{accountName}/users/register HTTP/1.1
Content-Type: application/json

  {
```

(continues on next page)

(continued from previous page)

```
    "username": "username_testermctesterson",
    "firstName": "Tester",
    "lastName": "McTesterton",
    "phoneNumber": "+15555555555",
    "emailAddress": "test@test.com",
    "securityQuestions": [
                            {
                               "question": "What would you say to this question?",
                               "answer": "mceasy123"
                            }
                         ],
    "newPassword": "newPassword"
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

newPassword [string, required] New user secret credentials for login.

firstName [string] First name of registering user.

lastName [string] Last name of registering user.

phoneNumber [string, required *if using phone verification*] Phone number of registering user.

emailAddress [string, required *if using email verification*] Email address of registering user.

securityQuestions [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

Responses

201 [Created]

- **New user has been registered and must be verified before use.**
  - This will only occur when Email or Text verification is enabled.

Example Result

```
{
   "username": "username_testermctesterson",
   "attempt": 1,
   "hash": "...",
   "expires": "1/1/1900",
   "hint": "..."
}
```

204 [No Content]

- **New user has been registered and is now available for use.**
  - This will only occur when Question verification is enabled.

400 [Bad request]

- Public registration is not enabled.
- Email address is required when Email recovery is enabled.
- Phone number is required when Text recovery is enabled.

- At least one Security Questions is required when Question recovery is enabled.
- Username is a required field.
- Email address must be in a valid format.
- Phone number must be in an international format.
- Username must be unique.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.6.4 New User

Creating a user is a controlled way where another user can grant access to someone else.

C#

```
 var client = MeshyClient.Initialize(accountName, publicKey);
 var connection = await client.LoginAnonymouslyAsync(username);

var user = new NewUser();

await connection.Users.CreateAsync(user);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

newPassword [string, required] New user secret credentials for login.

firstName [string] First name of registering user.

lastName [string] Last name of registering user.

phoneNumber [string, required *if using phone verification*] Phone number of registering user.

emailAddress [string, required *if using email verification*] Email address of registering user.

securityQuestions [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

verified [boolean, default: false] Identifies if the user is verified. The user must be verified to login if the verification method is email or phone number.

isActive [boolean, default: false] Identifies if the user is active. The user must be active to allow login.

roles [object] Collection of roles and when they were added to give user permissions within the system.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var user = await client.create({
                                username: username,
                                newPassword: newPassword,
                                firstName: firstName,
                                lastName: lastName,
                                phoneNumber: phoneNumber,
```

(continues on next page)

(continued from previous page)

```
                                    emailAddress: emailAddress,
                                    securityQuestions: securityQuestions,
                                    verified: verified,
                                    isActive: isActive,
                                    roles: roles
                                });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

newPassword [string, required] New user secret credentials for login.

firstName [string] First name of registering user.

lastName [string] Last name of registering user.

phoneNumber [string, required *if using phone verification*] Phone number of registering user.

emailAddress [string, required *if using email verification*] Email address of registering user.

securityQuestions [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

verified [boolean, default: false] Identifies if the user is verified. The user must be verified to login if the verification method is email or phone number.

isActive [boolean, default: false] Identifies if the user is active. The user must be active to allow login.

roles [object] Collection of roles and when they were added to give user permissions within the system.

REST

```
POST https://api.meshydb.com/{accountName}/users HTTP/1.1
Content-Type: application/json

  {
    "username": "username_testermctesterson",
    "firstName": "Tester",
    "lastName": "McTesterton",
    "phoneNumber": "+15555555555",
    "emailAddress": "test@test.com",
    "securityQuestions": [
                            {
                               "question": "What would you say to this question?",
                               "answer": "mceasy123"
                            }
                          ],
    "newPassword": "newPassword",
    verified: true,
    isActive: true,
    roles: []
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

newPassword [string, required] New user secret credentials for login.

firstName [string] First name of registering user.

lastName [string] Last name of registering user.

phoneNumber [string, required *if using phone verification*] Phone number of registering user.

emailAddress [string, required *if using email verification*] Email address of registering user.

securityQuestions [object[], required *if using question verification*] New set of questions and answers for registering user in password recovery.

verified [boolean, default: false] Identifies if the user is verified. The user must be verified to login if the verification method is email or phone number.

isActive [boolean, default: false] Identifies if the user is active. The user must be active to allow login.

roles [object] Collection of roles and when they were added to give user permissions within the system.

Responses

201 [Created]

- New user must be verified before use.

Example Result


```
{
   "username":"test",
   "firstName":null,
   "lastName":null,
   "verified":true,
   "isActive":true,
   "phoneNumber":null,
   "emailAddress":null,
   "roles":[
      {
         "name":"meshy.user",
         "addedDate":"2019-01-01T00:00:00.0000000+00:00"
      }
   ],
   "securityQuestions":[
      {
         "question":"test",
         "answerHash":"..."
      }
   ],
   "anonymous":false,
   "lastAccessed":null,
   "id":"5db..."
}
```


400 [Bad request]

- Email address is required when Email recovery is enabled.
- Phone number is required when Text recovery is enabled.
- At least one Security Questions is required when Question recovery is enabled.
- Username is a required field.
- Email address must be in a valid format.
- Phone number must be in an international format.

- Username must be unique.
- User cannot add roles they do not already have assigned. If a user has the update role permission they can assign any role to any user. However if they do not have this permission they can only assign roles they currently have assigned to themselves.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create users.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.6.5 Check Hash

Optionally, before verifying the request you can choose to check if the verification code provided is valid.

You may want to provide this flow if you still need to collect more information about the user before finalizing verification.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var check = new UserVerificationCheck();

var isValid = await client.CheckHashAsync(check);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, required] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string, required] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.checkHash({
                            username: username,
                            attempt: attempt:
                            hash: hash,
                            expires: expires,
                            hint: hint,
                            verificationCode: verificationCode
                        });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, required] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string, required] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

REST

```
POST https://api.meshydb.com/{accountName}/users/checkhash HTTP/1.1
Content-Type: application/json

  {
     "username": "username_testermctesterson",
     "attempt": 1,
     "hash": "...",
     "expires": "1/1/1900",
     "hint": "...",
     "verificationCode": "...",
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

attempt [integer, required] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string, required] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

Responses

200 [OK]

- Identifies if hash with verification code is valid.

Example Result

```
true
```

400 [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.6.6 Verify

If email or text verification is configured the registered user must be verified. The resulting request lasts one hour.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var check = new UserVerificationCheck();

await client.VerifyAsync(check);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, required] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string, required] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.verify({
                  username: username,
                  attempt: attempt:
                  hash: hash,
                  expires: expires,
                  hint: hint,
                  verificationCode: verificationCode
                                        });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, required] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string, required] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

REST

```
POST https://api.meshydb.com/{accountName}/users/verify HTTP/1.1
Content-Type: application/json

  {
     "username": "username_testermctesterson",
```

(continues on next page)

(continued from previous page)

```
    "attempt": 1,
    "hash": "...",
    "expires": "1/1/1900",
    "hint": "...",
    "verificationCode": "...",
}
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

attempt [integer, required] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string, required] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

Responses

204 [No Content]

- User has been verified successfully.

400 [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.
- Hash is expired.
- Anonymous user cannot be verified.
- User has already been verified.
- Request hash is invalid.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.7 Retrieving

Retrieving information about a user.

3.7.1 Self

Retrieve details about the authenticated user.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Users.GetSelfAsync();
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var self = await meshyConnection.usersService.getSelf();
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

REST

```
GET https://api.meshydb.com/{accountName}/users/me HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

Responses

200 [OK]

- Retrieves information about the authorized user.

Example Result

```
{
  "id": "5c78cc81dd870827a8e7b6c4",
  "username": "username_testermctesterson",
  "firstName": "Tester",
  "lastName": "McTesterton",
  "verified": true,
  "isActive": true,
  "phoneNumber": "5555555555",
  "roles" : [
              {
                 "name":"admin",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              },
              {
                 "name":"test",
```

(continues on next page)

(continued from previous page)

```
                "addedDate":"2019-01-01T00:00:00.0000000+00:00"
            }
        ],
  "securityQuestions": [
                          {
                             "question": "What would you say to this question?",
                             "answer": "..."
                          }
                       ],
  "anonymous": true,
  "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.7.2 Existing

Retrieve details about an existing user by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Users.GetAsync(id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

id [string, required] Identifies id of user.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var user = await meshyConnection.usersService.get(id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

id [string, required] Identifies id of user.

REST

```
GET https://api.meshydb.com/{accountName}/users/{id} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

id [string, required] Identifies id of user.

Responses

200 [OK]

- Retrieves information about the existing user.

Example Result

```
{
  "id": "5c78cc81dd870827a8e7b6c4",
  "username": "username_testermctesterson",
  "firstName": "Tester",
  "lastName": "McTesterton",
  "verified": true,
  "isActive": true,
  "phoneNumber": "5555555555",
  "roles" : [
              {
                 "name":"admin",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              },
              {
                 "name":"test",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              }
          ],
  "securityQuestions": [
                           {
                              "question": "What would you say to this question?",
                              "answer": "..."
                           }
                       ],
  "anonymous": true,
  "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read users.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.8 Searching

Filter User data based on query parameters.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Users.SearchAsync(name, roleId, orderBy, activeOnly, page, pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string] Case-insensitive partial name search.

roleId [string] Filters users with defined role identifier.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format.

activeOnly [boolean, default: true] Filters users that are not active. Setting to false will include all users regardless of active status.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

await meshyConnection.usersService.search({
                                            name: name,
                                            roleId: roleId,
                                            orderBy: orderBy,
                                            activeOnly: activeOnly,
                                            page: page,
                                            pageSize: pageSize
                                          });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string] Case-insensitive partial name search.

roleId [string] Filters users with defined role identifier.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format.

activeOnly [boolean, default: true] Filters users that are not active. Setting to false will include all users regardless of active status.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
 GET https://api.meshydb.com/{accountName}/users?name={name}&
                                                 roleId={roleId}&
                                                 orderBy={orderBy}&
                                                 activeOnly={activeOnly}&
                                                 page={page}&
                                                 pageSize={pageSize} HTTP/1.1
 Authorization: Bearer {access_token}

(Line breaks added for readability)
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

name [string] Case-insensitive partial name search.

roleId [string] Filters users with defined role identifier.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format.

activeOnly [boolean, default: true] Filters users that are not active. Setting to false will include all users regardless of active status.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Identifies if users were found.

Example Result


```
{
    "results":[
        {
            "username":"test",
            "firstName":null,
            "lastName":null,
            "verified":true,
            "isActive":true,
            "phoneNumber":null,
            "emailAddress":null,
            "roles":[
                {
                "name":"meshy.user",
                "addedDate":"2019-10-18T15:11:55.2413015-05:00"
                }
            ],
            "securityQuestions":[
                {
                "question":"Test 1",
                "answerHash":"..."
                }
```


(continues on next page)

(continued from previous page)

```
            ],
            "anonymous":false,
            "lastAccessed":null,
            "id":"5d4..."
        }
    ],
    "page":1,
    "pageSize":25,
    "totalRecords":1
}
```

400 [Bad request]

- User is not able to delete self.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read users.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.9 Updating

Updating self allows the ability to update the authenticated user's information.

This might be personal or security questions for password recovery later.

3.9.1 My Personal Information

The following can be used to update an authenticated user's personal information such as name, phone number, and email address.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var user = new User();

await connection.Users.UpdateSelfAsync(user);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

firstName [string] First name of authenticated user.

lastName [string] Last name of authenticated user.

phoneNumber [string, required *if using phone verification*] Phone number of authenticated user.

emailAddress [string, required *if using email verification*] Email address of authenticated user.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var self = await meshyConnection.usersService.updateSelf({
                                                    firstName: firstName,
                                                    lastName: lastName,
                                                    phoneNumber: phoneNumber,
                                                    emailAddress: emailAddress
                                                });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

firstName [string] First name of authenticated user.

lastName [string] Last name of authenticated user.

phoneNumber [string, required *if using phone verification*] Phone number of authenticated user.

emailAddress [string, required *if using email verification*] Email address of authenticated user.

REST

```
PUT https://api.meshydb.com/{accountName}/users/me HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

  {
    "firstName": "Tester",
    "lastName": "McTesterton",
    "phoneNumber": "+15555555555",
    "emailAddress": "test@test.com"
  }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

firstName [string] First name of authenticated user.

lastName [string] Last name of authenticated user.

phoneNumber [string, required *if using phone verification*] Phone number of authenticated user.

emailAddress [string, required *if using email verification*] Email address of authenticated user.

Responses

200 [OK] Updated information of updated authorized user.

Example Result

```
{
  "id": "5c78cc81dd870827a8e7b6c4",
  "username": "username_testermctesterson",
```

(continues on next page)

(continued from previous page)

```
  "firstName": "Tester",
  "lastName": "McTesterton",
  "verified": true,
  "isActive": true,
  "phoneNumber": "+15555555555",
  "emailAddress": "test@test.com",
  "roles" : [
              {
                 "name":"admin",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              },
              {
                 "name":"test",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              }
          ],
  "securityQuestions": [
                           {
                              "question": "What would you say to this question?",
                              "answer": "..."
                           }
                       ],
  "anonymous": false,
  "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```

400 [Bad request]

- Email address is required when Email recovery is enabled and the user is not anonymous.
- Phone number is required when Text recovery is enabled and the user is not anonymous.
- Username is a required field.
- Email address must be in a valid format.
- Phone number must be in an international format.
- Unable to change user roles via API.

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.9.2 Existing Personal Information

The following can be used to update an existing user's personal information such as name, phone number, and email address.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var user = new User();
```

(continues on next page)

(continued from previous page)

```
await connection.Users.UpdateAsync(id, user);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

id [string, required] Identifies id of user.

firstName [string] First name of authenticated user.

lastName [string] Last name of authenticated user.

phoneNumber [string, required *if using phone verification*] Phone number of authenticated user.

emailAddress [string, required *if using email verification*] Email address of authenticated user.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

var self = await meshyConnection.usersService.update(id,
                                                     {
                                                       firstName: firstName,
                                                       lastName: lastName,
                                                       phoneNumber: phoneNumber,
                                                       emailAddress: emailAddress
                                                     });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

id [string, required] Identifies id of user.

firstName [string] First name of authenticated user.

lastName [string] Last name of authenticated user.

phoneNumber [string, required *if using phone verification*] Phone number of authenticated user.

emailAddress [string, required *if using email verification*] Email address of authenticated user.

REST

```
PUT https://api.meshydb.com/{accountName}/users/{id} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

  {
    "firstName": "Tester",
    "lastName": "McTesterton",
    "phoneNumber": "+15555555555",
    "emailAddress": "test@test.com"
  }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

id [string, required] Identifies id of user.

firstName [string] First name of authenticated user.

lastName [string] Last name of authenticated user.

phoneNumber [string, required *if using phone verification*] Phone number of authenticated user.

emailAddress [string, required *if using email verification*] Email address of authenticated user.

Responses

200 [OK] Updated information of updated existing user.

Example Result


```
{
  "id": "5c78cc81dd870827a8e7b6c4",
  "username": "username_testermctesterson",
  "firstName": "Tester",
  "lastName": "McTesterton",
  "verified": true,
  "isActive": true,
  "phoneNumber": "+15555555555",
  "emailAddress": "test@test.com",
  "roles" : [
              {
                 "name":"admin",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              },
              {
                 "name":"test",
                 "addedDate":"2019-01-01T00:00:00.0000000+00:00"
              }
          ],
  "securityQuestions": [
                           {
                              "question": "What would you say to this question?",
                              "answer": "..."
                           }
                       ],
  "anonymous": false,
  "lastAccessed":"2019-01-01T00:00:00.0000+00:00"
}
```


400 [Bad request]

- Email address is required when Email recovery is enabled and the user is not anonymous.
- Phone number is required when Text recovery is enabled and the user is not anonymous.
- Username is a required field.
- Email address must be in a valid format.
- Phone number must be in an international format.
- Unable to change user roles via API.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update users.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.9.3 My Security Questions

The following can be used to change the authenticated user's security questions to be used for password recovery.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var questions = new UserSecurityQuestionUpdate();

questions.SecurityQuestions.Add(new SecurityQuestion(){
                                                        Question = "What should␣
↪this be?",
                                                        Answer = "This seems like␣
↪an ok example"
                                                    };

await connection.Users.UpdateSecurityQuestionsAsync(questions);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

securityQuestions [object[], required] New set of questions and answers for authenticated user in password recovery.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username, password);

await meshyConnection.usersService.updateSecurityQuestion({
                                                            securityQuestions:␣
↪securityQuestions
                                                        });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

securityQuestions [object[], required] Collection of questions and answers used for password recovery if question security is configured.

REST

```
POST https://api.meshydb.com/{accountName}/users/me/questions HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

  {
    "securityQuestions": [
                            {
                                "question": "What would you say to this question?",
```

(continues on next page)

(continued from previous page)

```
                                "answer": "..."
                            }
                        ]
    }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

securityQuestions [object[], required] New set of questions and answers for authenticated user in password recovery.

Responses

204 [No Content]

- Updated information of updated authorized user.

400 [Bad request]

- Unable to update security questions if question verification is not configured.
- Anonymous user cannot have security questions.
- At least one question is required.
- Question text is required.
- Answer is required.

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.9.4 Existing Security Questions

The following can be used to change the authenticated user's security questions to be used for password recovery.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var questions = new UserSecurityQuestionUpdate();

questions.SecurityQuestions.Add(new SecurityQuestion(){
                                                        Question = "What should␣
↪this be?",
                                                        Answer = "This seems like␣
↪an ok example"
                                                    };

await connection.Users.UpdateSecurityQuestionsAsync(id, questions);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

id [string, required] Identifies id of user.

securityQuestions [object[], required] New set of questions and answers for authenticated user in password recovery.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username, password);

await meshyConnection.usersService.updateUserSecurityQuestion(id,
                                                               {
                                                                 securityQuestions:␣
↪securityQuestions
                                                               });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

id [string, required] Identifies id of user.

securityQuestions [object[], required] Collection of questions and answers used for password recovery if question security is configured.

REST

```
POST https://api.meshydb.com/{accountName}/users/{id}/questions HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

  {
    "securityQuestions": [
                           {
                               "question": "What would you say to this question?",
                               "answer": "..."
                           }
                         ]
  }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

id [string, required] Identifies id of user.

securityQuestions [object[], required] New set of questions and answers for authenticated user in password recovery.

Responses

204 [No Content]

- Updated information of updated existing user.

400 [Bad request]

- Unable to update security questions if question verification is not configured.
- Anonymous user cannot have security questions.
- At least one question is required.
- Question text is required.

- Answer is required.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update users.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.9.5 Changing Password

Allows the authenticated user to change their password.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginWithPasswordAsync(username, password);

await connection.UpdatePasswordAsync(previousPassword, newPassword);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

previousPassword [string, required] Previous user secret credentials for login.

newPassword [string, required] New user secret credentials for login.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.login(username, password);

await meshyConnection.updatePassword(previousPassword, newPassword);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

password [string, required] User secret credentials for login. When anonymous it is static as nopassword.

previousPassword [string, required] Previous user secret credentials for login.

newPassword [string, required] New user secret credentials for login.

REST

```
POST https://api.meshydb.com/{accountName}/users/me/password HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{
  "newPassword": "newPassword",
  "previousPassword": "previousPassword"
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token: string, required Token identifying authorization with MeshyDB requested during Generate Access Token.

previousPassword [string, required] Previous user secret credentials for login.

newPassword [string, required] New user secret credentials for login.

Responses

204 [No Content]

- Identifies password was updated successfully.

400 [Bad request]

- Anonymous user cannot change password.
- New password is required.
- Previous password is required.
- Previous password does not match existing password.

401 [Unauthorized]

- User is not authorized to make call.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.10 Password Recovery

When a previously authenticated user used the system, they may need to recover their password. The request will be valid for one hour.

They will first need to request a forgot password before they are able to reset it.

Depending on your verification flow, whether it be email, text or security questions the user will need to either provide a code or answer to question to prove their knowledge of the request.

3.10.1 Forgetting Password

Creates a request for password reset that must have the matching data to reset to ensure request parity.

If using security questions, you can provide an attempt number to select which question is used for verification.

The attempt will load the question into the hint field to be asked of the user.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

await client.ForgotPasswordAsync(username, attempt);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies how many times a request has been made.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.forgotPassword(username, attempt);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies how many times a request has been made.

REST

```
POST https://api.meshydb.com/{accountName}/users/forgotpassword HTTP/1.1
Content-Type: application/json

  {
    "username": "username_testermctesterson",
    "attempt": 1
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies how many times a request has been made.

Responses

200 [OK]

- Generates forgot password response to be used for password reset.

Example Result

```
{
        "username": "username_testermctesterson",
        "attempt": 1,
        "hash": "...",
        "expires": "1900-01-01T00:00:00.000Z",
        "hint": "..."
}
```

400 [Bad request]

- Username is required.

- Anonymous user cannot recover password.

404 [Not Found]

- User was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.10.2 Check Hash

Optionally, before the user's password is reset you can check if the verification code, they provide is valid.

This would allow a user to verify their code before requiring a reset.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var check = new UserVerificationCheck();

var isValid = await client.CheckHashAsync(check);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

await client.checkHash({
                            username: username,
                            attempt: attempt,
                            hash: hash,
                            expires: expires,
                            verificationCode: verificationCode
                        });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

REST

```
POST https://api.meshydb.com/{accountName}/users/checkhash HTTP/1.1
Content-Type: application/json

  {
     "username": "username_testermctesterson",
     "attempt": 1,
     "hash": "...",
     "expires": "1900-01-01T00:00:00.000Z",
     "verificationCode": "..."
  }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

Responses

200 [OK]

- Identifies if hash with verification code is valid.

Example Result

```
true
```

400 [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.10.3 Resetting Password

Take result from forgot password and application verification code generated from email/text or security question answer, along with a new password to be used for login.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);

var passwordResetHash = await client.ForgotPasswordAsync(username, attempt);

var resetPassword = new ResetPassword() {
                                          Username = passwordResetHash.Username,
                                          Attempt = passwordResetHash.Attempt,
                                          Hash = passwordResetHash.Hash,
                                          Expires = passwordResetHash.Expires,
                                          VerificationCode = verificationCode,
                                          NewPassword = newPassword
                                        };

await client.ResetPasswordAsync(resetPassword);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

newPassword [string, required] New user secret credentials for login.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var passwordResetHash = await client.forgotPassword(username);

await client.resetPassword({
                             username: passwordResetHash.username,
                             attempt: passwordResetHash.attempt,
                             hash: passwordResetHash.hash,
                             expires: passwordResetHash.expires,
                             verificationCode: verificationCode,
                             newPassword: newPassword
                           });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

newPassword [string, required] New user secret credentials for login.

REST

```
POST https://api.meshydb.com/{accountName}/users/resetpassword HTTP/1.1
Content-Type: application/json

   {
      username: "username_testermctesterson",
      attempt: 1,
      hash: "...",
      expires: "1900-01-01T00:00:00.000Z",
      verificationCode: "...",
      newPassword: "..."
   }
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

attempt [integer, default: 1] Identifies which attempt hash was generated against.

hash [string, required] Generated hash from verification request.

expires [date, required] Identifies when the request expires.

hint [string] Hint for verification code was generated.

verificationCode [string, required] Value to verify against verification request.

newPassword [string, required] New user secret credentials for login.

Responses

204 [No Content]

- Identifies password reset is successful.

400 [Bad request]

- Username is required.
- Hash is required.
- Expires is required.
- Verification code is required.
- Hash is expired.
- New password is required.
- Anonymous user cannot be reset.
- User has already been verified.
- Request hash is invalid.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.11 Logging Out

Log authenticated user out.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.SignoutAsync();
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

await meshyConnection.signout();
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

REST

```
POST https://api.meshydb.com/{accountName}/connect/revocation HTTP/1.1
Content-Type: application/x-www-form-urlencoded

  token={refresh_token}&
  token_type_hint=refresh_token&
  client_id={publicKey}

(Form-encoding removed, and line breaks added for readability)
```

accountName [string, required] Indicates which account you are connecting to.

refresh_token [string, required] Refresh token identifying authorization with MeshyDB requested during Generating Token.

publicKey [string, required] Public identifier of connecting service.

Responses

200 [OK]

- Identifies successful logout.

400 [Bad request]

- Invalid client id.

- Token is missing.
- Unsupported Token type.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.12 Deleting

Remove user from system permanently.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Users.DeleteAsync(id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

id [string, required] Identifies id of user.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.usersService.delete(id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

id [string, required] Identifies id of user.

REST

```
DELETE https://api.meshydb.com/{accountName}/users/{id} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

id [string, required] Identifies id of user.

Responses

204 [No Content]

- Identifies if user was deleted.

400 [Bad request]

- User is not able to delete self.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete users.

404 [Not Found]

- User is not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.13 Creating

Create new custom mesh data into specified mesh name.

3.13.1 Single

Create a specific record.

C#

```
// Mesh is derived from class name
public class Person: MeshData
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
}

var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var person = await connection.Meshes.CreateAsync(new Person(){
  FirstName="Bob",
  LastName="Bobberson"
});
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, default: class name] Identifies name of mesh collection. e.g. person.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);
```

(continues on next page)

(continued from previous page)

```
var createdMesh = await meshyConnection.meshes.create(meshName,
                                                      {
                                                         firstName:"Bob",
                                                         lastName:"Bobberson"
                                                      });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required] Identifies name of mesh collection. e.g. person.

REST

```
POST https://api.meshydb.com/{accountName}/meshes/{mesh} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

   {
      "firstName": "Bob",
      "lastName": "Bobberson"
   }
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

Responses

201 [Created]

- Result of newly created mesh data.

Example Result

```
{
   "_id": "5c78cc81dd870827a8e7b6c4",
   "firstName": "Bob",
   "lastName": "Bobberson"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.
- Mesh already exists for provided id.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create meshes or specific mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.13.2 Many

Bulk create many objects.

C#

```
// Mesh is derived from class name
public class Person: MeshData
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
}

var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);
var data = new List<Person>();

data.Add(new Person(){
  FirstName="Bob",
  LastName="Bobberson"
});

var result = await connection.Meshes.CreateMany(data);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, default: class name] Identifies name of mesh collection. e.g. person.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
var anonymousUser = await client.registerAnonymousUser();
var connection = await client.loginAnonymously(anonymousUser.username);

var result = await connection.meshesService.createMany(meshName, [{
                                                                    firstName:"Bob",
                                                                    lastName:
↪"Bobberson"
                                                                }]);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required] Identifies name of mesh collection. e.g. person.

REST

```
POST https://api.meshydb.com/{accountName}/meshes/{mesh} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
    [{
        "firstName": "Bob",
        "lastName": "Bobberson"
    }]
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

Responses

201 [Created]

- Result of newly created mesh data.

Example Result

```
{
    "createdCount": 1
}
```

400 [Bad request]

- No data was provided.
- **Data is in an invalid format. The status of each object will be brought back to identify the error. The errors are as fol**
  - Mesh name is invalid and must be alpha characters only.
  - Mesh property cannot begin with '$' or contain '.'.
  - Mesh already exists for provided id.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create meshes or specific mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.14 Retrieving

Retrieve single item from Mesh collection.

C#

```
// Mesh is derived from class name
public class Person: MeshData
{
  public string FirstName { get; set; }
  public string LastName { get; set; }
```

(continues on next page)

(continued from previous page)

```
}

var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var person = await connection.Meshes.GetData<Person>(id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, default: class name] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies location of what Mesh data to retrieve.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var meshData = await meshyConnection.meshes.get(meshName, id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies location of what Mesh data to retrieve.

REST

```
GET https://api.meshydb.com/{accountName}/meshes/{mesh}/{id} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies location of what Mesh data to retrieve.

Responses

200 [OK]

- Mesh data found with given identifier.

Example Result

```
{
   "_id":"5c78cc81dd870827a8e7b6c4",
   "firstName": "Bob",
   "lastName": "Bobberson"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read meshes or mesh.

404 [Not Found]

- Mesh data was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.15 Updating

Update Mesh data in collection.

3.15.1 Single

Update single mesh data record.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);
var person = await connection.Meshes.GetDataAsync<Person>(id);

person.FirstName = "Bobbo";

person = await connection.Meshes.UpdateAsync(person);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, required, default: class name] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to replace.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();
```

(continues on next page)

(continued from previous page)

```
var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var meshData = await meshyConnection.meshes.update(meshName,
                                                   {
                                                      firstName:"Bob",
                                                      lastName:"Bobberson"
                                                   },
                                                   id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to replace.

REST

```
PUT https://api.meshydb.com/{accountName}/meshes/{mesh}/{id} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
   "firstName": "Bobbo",
   "lastName": "Bobberson"
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to replace.

Responses

200 [OK]

- Result of updated mesh data.

Example Result

```
{
   "_id":"5c78cc81dd870827a8e7b6c4",
   "firstName": "Bobbo",
   "lastName": "Bobberson"
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Mesh property cannot begin with '$' or contain '.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.15.2 Many

Bulk update data based on provided filter.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var result = await connection.Meshes.UpdateManyAsync<Person>(filter, update);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, required, default: class name] Identifies name of mesh collection. e.g. person.

filter [string, required] Criteria provided in a MongoDB format to limit results.

update [string, required] Update command provided in a MongoDB format.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
var anonymousUser = await client.registerAnonymousUser();
var connection = await client.loginAnonymously(anonymousUser.username);

var result = await connection.meshesService.updateMany(meshName, filter, update);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required] Identifies name of mesh collection. e.g. person.

filter [string, required] Criteria provided in a MongoDB format to limit results.

update [string, required] Update command provided in a MongoDB format.

REST

```
PATCH https://api.meshydb.com/{accountName}/meshes/{mesh} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
   "filter": filter,
   "update": update
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

filter [string, required] Criteria provided in a MongoDB format to limit results.

update [string, required] Update command provided in a MongoDB format.

Responses

200 [OK]

- Result of updated mesh data.

Example Result

```
{
   "isAcknowledged": true,
   "isModifiedCountAvailable": true,
   "matchedCount": 5,
   "modifiedCount": 3,
   "upsertedId": null
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is required.
- Update is required.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Update is in an invalid format. It must be in a valid Mongo DB format.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.16 Searching

Filter Mesh data from collection based on query parameters.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var pagedPersonResult = await connection.Meshes.SearchAsync<Person>(filter, page,␣
↪pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, required, default: class name] Identifies name of mesh collection. e.g. person.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var pagedResults = await meshyConnection.meshes.search(meshName,
                                                       {
                                                          filter: filter,
                                                          orderBy: orderBy,
                                                          pageNumber: page,
                                                          pageSize: pageSize
                                                       });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

meshName [string, required] Identifies name of mesh collection. e.g. person.

username [string] Unique user name for authentication.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
GET https://api.meshydb.com/{accountName}/meshes/{mesh}?filter={filter}&
                                                       orderBy={orderBy}&
                                                       page={page}&
                                                       pageSize={pageSize} HTTP/1.1
Authorization: Bearer {access_token}
```

(Line breaks added for readability)

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Mesh data found with given search criteria.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results": [{
                   "_id":"5c78cc81dd870827a8e7b6c4",
                   "firstName": "Bobbo",
                   "lastName": "Bobberson"
               }],
   "totalRecords": 1
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter is in an invalid format. It must be in a valid Mongo DB format.
- Order by is in an invalid format. It must be in a valid Mongo DB format.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.17 Deleting

Delete mesh data from a collection is permanent. If you desire to retain your data one possible pattern to consider would be a soft-delete pattern.

3.17.1 Single

Delete a specific record of data.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Meshes.DeleteAsync(person);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required, default: class name] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to remove.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

await meshyConnection.meshes.delete(meshName, id);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

mesh [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to remove.

REST

```
DELETE https://api.meshydb.com/{accountName}/meshes/{mesh}/{id} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

id [string, required] Identifies unique record of Mesh data to remove.

Responses

204 [No Content]

- Mesh has been deleted successfully.

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete meshes or mesh.

404 [Not Found]

- Mesh data was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.17.2 Many

Delete all objects with the provided filter.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var result = connection.Meshes.DeleteMany<DeleteManyTest>(filter);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required, default: class name] Identifies name of mesh collection. e.g. person.

filter [string, required] Criteria provided in a MongoDB format to limit results.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
var anonymousUser = await client.registerAnonymousUser();
var connection = await client.loginAnonymously(anonymousUser.username);

var data = await connection.meshesService.deleteMany(meshName, filter)
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

meshName [string, required, default: class name] Identifies name of mesh collection. e.g. person.

filter [object, required] Criteria provided in a MongoDB format to limit results.

REST

```
DELETE https://api.meshydb.com/{accountName}/meshes/{mesh}?filter={filter} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

filter [string, required] Criteria provided in a MongoDB format to limit results.

Responses

200 [OK]

- Mesh data found with given search criteria and successfully deleted.

Example Result

```
{
   "deletedCount": 5,
   "isAcknowledged": true
}
```

400 [Bad request]

- Mesh name is invalid and must be alpha characters only.
- Filter was not provided.
- Filter is in an invalid format.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete meshes or mesh.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.18 Projecting

A projection is a stored MongoDB query that can contain aggregations, filtering and lookups. It's great for reporting or simply re-using queries throughout your application. Projections also give you the ability to control access to data by allowing you to explicitly define the data you want to make accessible to end-users.

You can create your query your Mesh Data using the admin portal under the Projections tab.

Please review the following for supported *aggregations*.

3.18.1 API Reference

C#

```
public class PopularState
{
   [JsonProperty("state")]
   public string State { get; set; }

   [JsonProperty("attractions")]
   public int Attractions { get; set; }
}

var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);
```

(continues on next page)

(continued from previous page)

```
var stateAttractions = await connection.Projections.Get<PopularState>(projectionName,
                                                                      filter,
                                                                      orderBy,
                                                                      page,
                                                                      pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

projectionName [string, required] Identifies name of mesh collection. e.g. person.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [object] Defines which fields need to be ordered and direction. Review more ways to use *ordering*.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var anonymousUser = await client.registerAnonymousUser();

var meshyConnection = await client.loginAnonymously(anonymousUser.username);

var popularStates = await meshyConnection.projections.get<any>(projectionName,
                                                               {
                                                                   filter: filter,
                                                                   orderBy: orderBy,
                                                                   page: page,
                                                                   pageSize: pageSize
                                                               });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

projectionName [string, required] Identifies name of mesh collection. e.g. person.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format. Review more ways to use *ordering*.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
GET https://api.meshydb.com/{accountName}/projections/{projectionName} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

mesh [string, required] Identifies name of mesh collection. e.g. person.

projectionName [string, required] Identifies name of mesh collection. e.g. person.

filter [string] Criteria provided in a MongoDB format to limit results.

orderBy [string] Defines which fields need to be ordered and direction in a MongoDB format. Review more ways to use *ordering*.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Projection found with given identifier.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results": [{
                   "state":"WI",
                   "attractions":"24"
               }],
   "totalRecords": 1
}
```

400 [Bad request]

- Projection name is required.
- Order by is invalid.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read projections.

404 [Not Found]

- Projection was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.18.2 Ordering Data

Ordering is supported in a MongoDB format. This format is as an object with a -1 or 1 to identify descending or ascending format respectively.

The following example shows how to sort an object by Name in descending order.

C#

```
var orderBy = OrderByDefinition<PopularState>.OrderByDescending("Name");

// Or

orderBy = OrderByDefinition<PopularState>.OrderByDescending(x => x.Name);

var popularStates = await connection.Projections.Get<PopularState>(projectionName,
                                                                   orderBy,
                                                                   page,
                                                                   pageSize);
```

Alternatively you can use MongoDB syntax.

```
var orderBy = "{ \"Name\": -1 }";

var popularStates = await connection.Projections.Get<PopularState>(projectionName,
                                                                   orderBy,
                                                                   page,
                                                                   pageSize);
```

NodeJS

```
var orderBy = { "Name": -1 };

var popularStates = await meshyConnection.projections.get<any>(projectionName,
                                                               {
                                                                   orderBy: orderBy,
                                                                   page: page,
                                                                   pageSize: pageSize
                                                               });
```

REST

```
GET https://api.meshydb.com/{accountName}/projections/{projectionName}?orderBy={ "Name
↪": -1 } HTTP/1.1
Authorization: Bearer {access_token}
```

Additional filters can be extended as follows.

This example will order by Name descending then Age ascending.

C#

```
var orderBy = OrderByDefinition<Person>.OrderByDescending("Name").ThenBy("Age");

// Or

orderBy = OrderByDefinition<Person>.OrderByDescending(x => x.Name).ThenBy(x=> x.Age);

var popularStates = await connection.Projections.Get<PopularState>(projectionName,
                                                                   orderBy,
                                                                   page,
                                                                   pageSize);
```

Alternatively you can use MongoDB syntax

```
var orderBy = "{ \"Name\": -1, \"Age\": 1 }";
```

(continues on next page)

(continued from previous page)

```
var popularStates = await connection.Projections.Get<PopularState>(projectionName,
                                                                   orderBy,
                                                                   page,
                                                                   pageSize);
```

NodeJS

```
var orderBy = { "Name": -1, "Age": 1 };

var popularStates = await meshyConnection.projections.get<any>(projectionName,
                                                               {
                                                                     orderBy: orderBy,
                                                                     page: page,
                                                                     pageSize: pageSize
                                                               });
```

REST

```
GET https://api.meshydb.com/{accountName}/projections/{projectionName}?orderBy={ "Name
↪": -1, "Age": 1 } HTTP/1.1
Authorization: Bearer {access_token}
```

3.18.3 Supported Aggregates

The following aggregates are from MongoDB and more detailed explanation can be found here.

- $addFields
- $bucket
- $bucketAuto
- $count
- $graphLookup
- $facet
- $group
- $limit
- $lookup
- $match
- $project
- $redact
- $replaceRoot
- $sample
- $skip
- $sort
- $sortByCount
- $unwind

3.19 Creating

Roles allow the ability to setup specific permissions for users.

These permissions can either be set as a blanket allow for data or be granlar to specific data sets.

3.19.1 Role

Creating a role allows it to be assignable to a user.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var role = new Role();

await connection.Roles.CreateAsync(role);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string, required] Name of the role.

description [string] Describes the purpose of the role.

numberOfUsers [string] Read-only count of users assigned to the role.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.create(role);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string, required] Name of the role.

description [string] Describes the purpose of the role.

numberOfUsers [string] Read-only count of users assigned to the role.

REST

```
POST https://api.meshydb.com/{accountName}/roles HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
    "name":"test",
    "description":"..."
}
```

accountName [string, required] Indicates which account you are connecting to.

username [string, required] Unique user name for authentication.

name [string, required] Name of the role.

description [string] Describes the purpose of the role.

Responses

201 [Created]

- Identifies if role was created.

Example Result

```
{
    "name":"test",
    "description":"...",
    "id":"5db..."
}
```

400 [Bad request]

- Name is required.
- Name can only be alpha characters only.
- Role cannot start with 'meshy.'.
- Role already exists.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create roles.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.19.2 Permission

When creating a permission it is assigned to a role. When a user has the role this permission will take effect on their next signin/token refresh.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var permission = new Permission();

await connection.Roles.CreatePermissionAsync(roleId, permission);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

permissibleName [string, required] Name of permissible reference. An example would be 'meshes' or 'meshes.{meshName}' to identify access to a specific mesh.

create [type:*boolean*] Identifies if role can create data.

update [type:*boolean*] Identifies if role can update data.

read [type:*boolean*] Identifies if role can read data.

delete [type:*boolean*] Identifies if role can delete data.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);
var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.createPermission(roleId, permission);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

permissibleName [string, required] Name of permissible reference. An example would be 'meshes' or 'meshes.{meshName}' to identify access to a specific mesh.

create [type:*boolean*] Identifies if role can create data.

update [type:*boolean*] Identifies if role can update data.

read [type:*boolean*] Identifies if role can read data.

delete [type:*boolean*] Identifies if role can delete data.

REST

```
POST https://api.meshydb.com/{accountName}/roles/{roleId}/permissions HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
    "permissibleName":"meshes",
    "create":"true",
    "update":"true",
    "read":"true",
    "delete":"true"
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

permissibleName [string, required] Name of permissible reference. An example would be 'meshes' or 'meshes.{meshName}' to identify access to a specific mesh.

create [type:*boolean*] Identifies if role can create data.

update [type:*boolean*] Identifies if role can update data.

read [type:*boolean*] Identifies if role can read data.

delete [type:*boolean*] Identifies if role can delete data.

Responses

201 [Created]

- Identifies if role was created.

Example Result

```
{
    "id":"5db...",
    "permissibleName":"meshes",
    "create":"true",
    "update":"true",
    "read":"true",
    "delete":"true"
}
```

400 [Bad request]

- Permissible name is required.
- At least one of the following must be set: Create, Update, Read, Delete.
- Permissible does not exist.
- Permisisble does not support the permission configuration.
- Role does not exist.
- Permissible was already configured for role.
- A higher permissible cannot be assigned to a role with a specific permission already. IE you cannot have 'meshes' and 'meshes.person' for the role.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to create permissions.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.20 Retrieving

Retrieve information for specific role.

3.20.1 Role

Retrieve details about role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.GetAsync(roleId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.get(roleId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

REST

```
GET https://api.meshydb.com/{accountName}/roles/{roleId} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

Responses

200 [OK]

- Identifies if role was found.

Example Result

```
{
    "name":"test",
    "description":"...",
    "id":"5db..."
}
```

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read roles.

404 [Not Found]

- Role was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.20.2 Permission

Get specific permission from role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var permission = await connection.Roles.GetPermissionAsync(roleId, permissionId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

var permission = await meshyConnection.rolesService.getPermission(roleId,␣
↪permissionId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

REST

```
GET https://api.meshydb.com/{accountName}/roles/{roleId}/permissions/{permissionId}␣
↪HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

Responses

200 [OK]

- Identifies if permission was found.

Example Result

```
{
    "id":"5db...",
    "permissibleName":"meshes",
    "create":"true",
    "update":"true",
    "read":"true",
    "delete":"true"
}
```

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read roles.

404 [Not Found]

- Permission was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.21 Updating

Update information for specific role.

3.21.1 Role

Update details about role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.UpdateAsync(roleId, role);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

name [string, required] Name of the role.

description [string] Describes the purpose of the role.

numberOfUsers [string] Read-only count of users assigned to the role.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.update(roleId, role);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

REST

```
PUT https://api.meshydb.com/{accountName}/roles/{roleId} HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
    "name":"test",
    "description":"..."
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

Responses

200 [OK]

- Identifies if role was updated.

Example Result

```
{
    "name":"test",
    "description":"...",
    "id":"5db..."
}
```

400 [Bad request]

- Name is required.
- Name can only be alpha characters only.
- Role cannot start with 'meshy.'.
- Role already exists.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update roles.

404 [Not Found]

- Role was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.21.2 Permission

Update specific permission from role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.UpdatePermissionAsync(roleId, permissionId, permission);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

permissibleName [string, required] Name of permissible reference. An example would be 'meshes' or 'meshes.{meshName}' to identify access to a specific mesh.

create [type:*boolean*] Identifies if role can create data.

update [type:*boolean*] Identifies if role can update data.

read [type:*boolean*] Identifies if role can read data.

delete [type:*boolean*] Identifies if role can delete data.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.deletePermission(roleId, permissionId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

permissibleName [string, required] Name of permissible reference. An example would be 'meshes' or 'meshes.{meshName}' to identify access to a specific mesh.

create [type:*boolean*] Identifies if role can create data.

update [type:*boolean*] Identifies if role can update data.

read [type:*boolean*] Identifies if role can read data.

delete [type:*boolean*] Identifies if role can delete data.

REST

```
PUT https://api.meshydb.com/{accountName}/roles/{roleId}/permissions/{permissionId}␣
↪HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
    "permissibleName":"meshes",
    "create":"true",
    "update":"true",
    "read":"true",
    "delete":"true"
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

permissibleName [string, required] Name of permissible reference. An example would be 'meshes' or 'meshes.{meshName}' to identify access to a specific mesh.

create [type:*boolean*] Identifies if role can create data.

update [type:*boolean*] Identifies if role can update data.

read [type:*boolean*] Identifies if role can read data.

delete [type:*boolean*] Identifies if role can delete data.

Responses

200 [OK]

- Identifies if permission was updated.

Example Result

```
{
    "id":"5db...",
    "permissibleName":"meshes",
    "create":"true",
    "update":"true",
    "read":"true",
    "delete":"true"
}
```

400 [Bad request]

- Permissible name is required.
- At least one of the following must be set: Create, Update, Read, Delete.
- Permissible does not exist.
- Permisisble does not support the permission configuration.
- Role does not exist.

- Permissible was already configured for role.
- A higher permissible cannot be assigned to a role with a specific permission already. IE you cannot have 'meshes' and 'meshes.person' for the role.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update roles.

404 [Not Found]

- Permission was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.21.3 Add Users

Add users from specific role.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var batchRoleAdd = new BatchRoleAdd();

await connection.Roles.AddUsersAsync(roleId, batchRoleAdd);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

batchRoleAdd [object, required] Batch object of user ids to be added from role.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.addUsers(roleId, batchRoleAdd);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

batchRoleAdd [object, required] Batch object of user ids to be added from role.

REST

```
POST https://api.meshydb.com/{accountName}/roles/{roleId}/users HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
    "users": [
                {
                    "id":"5db..."
                }
            ]
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

Responses

204 [No Content]

- Identifies if role is added.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update roles.

404 [Not Found]

- Role was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.22 Searching

Search for information related to roles.

3.22.1 Role

Search details about roles.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.SearchAsync(name, page, pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string] Case-insensitive partial name search.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.search({
                                            name: name,
                                            page: page,
                                            pageSize: pageSize
                                          });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string] Case-insensitive partial name search.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
GET https://api.meshydb.com/{accountName}/roles?name={name}&
                                                page={page}&
                                                pageSize={pageSize} HTTP/1.1
Authorization: Bearer {access_token}

(Line breaks added for readability)
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

name [string] Case-insensitive partial name search.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Identifies if role were found.

Example Result

```
{
   "page": 1,
   "pageSize": 25,
   "results": [{
```

(continues on next page)

(continued from previous page)

```
                    "name":"test",
                    "description":"...",
                    "id":"5db...",
                    "numberOfUsers": 0
                }],
    "totalRecords": 1
}
```

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read roles.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.22.2 Permission

Search permissions from role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.SearchPermissionAsync(roleId, permissibleName, page, pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

permissibleName [string] Case-insensitive partial name search of permissible.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.searchPermission(roleId, {
                                                        permissibleName:␣
↪permissibleName,
                                                        page: page,
                                                        pageSize: pageSize
                                                    });
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

permissibleName [string] Case-insensitive partial name search of permissible.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
GET https://api.meshydb.com/{accountName}/roles/{roleId}/permissions?permissibleName=
↪{permissibleName}&
                                                                    page={page}&
                                                                    pageSize=
↪{pageSize} HTTP/1.1
Authorization: Bearer {access_token}

(Line breaks added for readability)
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

permissibleName [string] Case-insensitive partial name search of permissible.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Identifies if permissions were found.

Example Result

```
{
    "results":[
        {
            "permissibleName":"meshes",
            "create":true,
            "update":true,
            "read":true,
            "delete":true,
            "id":"5d9..."
        }
    ],
    "page":1,
    "pageSize":25,
    "totalRecords":1
}
```

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read roles.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.22.3 Permissibles

Search for permissible to assign to a permission.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.SearchPermissibleAsync(name, page, pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

name [string] Case-insensitive partial name search.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.searchPermissible(name, page, pageSize);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

name [string] Case-insensitive partial name search.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

REST

```
GET https://api.meshydb.com/{accountName}/permissibles?name={name}&
                                                       page={page}&
                                                       pageSize={pageSize} HTTP/1.1
Authorization: Bearer {access_token}

(Line breaks added for readability)
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

name [string] Case-insensitive partial name search.

page [integer, default: 1] Page number of results to bring back.

pageSize [integer, max: 200, default: 25] Number of results to bring back per page.

Responses

200 [OK]

- Identifies if permissibles were found.

Example Result

```
{
    "results":[
        {
            "name":"meshes",
            "canCreate":true,
            "canUpdate":true,
            "canRead":true,
            "canDelete":true
        }
    ],
    "page":1,
    "pageSize":25,
    "totalRecords":1
}
```

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to read roles.

404 [Not Found]

- Role was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.23 Deleting

Delete information for specific role.

3.23.1 Role

Delete details about role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.DeleteAsync(roleId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.delete(roleId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

REST

```
DELETE https://api.meshydb.com/{accountName}/roles/{roleId} HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

Responses

204 [No Content]

- Identifies if role was deleted.

400 [Bad request]

- Unable to delete role that starts with 'meshy.'.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete roles.

404 [Not Found]

- Role was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.23.2 Permission

Delete specific permission from role by id.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

await connection.Roles.DeletePermissionAsync(roleId, permissionId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.deletePermission(roleId, permissionId);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

REST

```
DELETE https://api.meshydb.com/{accountName}/roles/{roleId}/permissions/{permissionId}
↪ HTTP/1.1
Authorization: Bearer {access_token}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

permissionId [string, required] Identifies id of permission.

Responses

204 [No Content]

- Identifies if permission was deleted.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to delete roles.

404 [Not Found]

- Permission was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.

3.23.3 Remove Users

Remove users from specific role.

C#

```
var client = MeshyClient.Initialize(accountName, publicKey);
var connection = await client.LoginAnonymouslyAsync(username);

var batchRoleRemove = new BatchRoleRemove();

await connection.Roles.RemoveUsersAsync(roleId, batchRoleRemove);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

roleId [string, required] Identifies id of role.

batchRoleRemove [object, required] Batch object of user ids to be removed from role.

NodeJS

```
var client = MeshyClient.initialize(accountName, publicKey);

var meshyConnection = await client.loginAnonymously(username);

await meshyConnection.rolesService.removeUsers(roleId, batchRoleRemove);
```

accountName [string, required] Indicates which account you are connecting to.

publicKey [string, required] Public identifier of connecting service.

username [string, required] Unique user name for authentication.

roleId [string, required] Identifies id of role.

batchRoleRemove [object, required] Batch object of user ids to be removed from role.

REST

```
DELETE https://api.meshydb.com/{accountName}/roles/{roleId}/users HTTP/1.1
Authorization: Bearer {access_token}
Content-Type: application/json

{
    "users": [
                {
                    "id":"5db..."
                }
            ]
}
```

accountName [string, required] Indicates which account you are connecting to.

access_token [string, required] Token identifying authorization with MeshyDB requested during Generating Token.

roleId [string, required] Identifies id of role.

Responses

204 [No Content]

- Identifies if role is removed.

401 [Unauthorized]

- User is not authorized to make call.

403 [Forbidden]

- User has insufficent permission to update roles.

404 [Not Found]

- Role was not found.

429 [Too many request]

- You have either hit your API or Database limit. Please review your account.