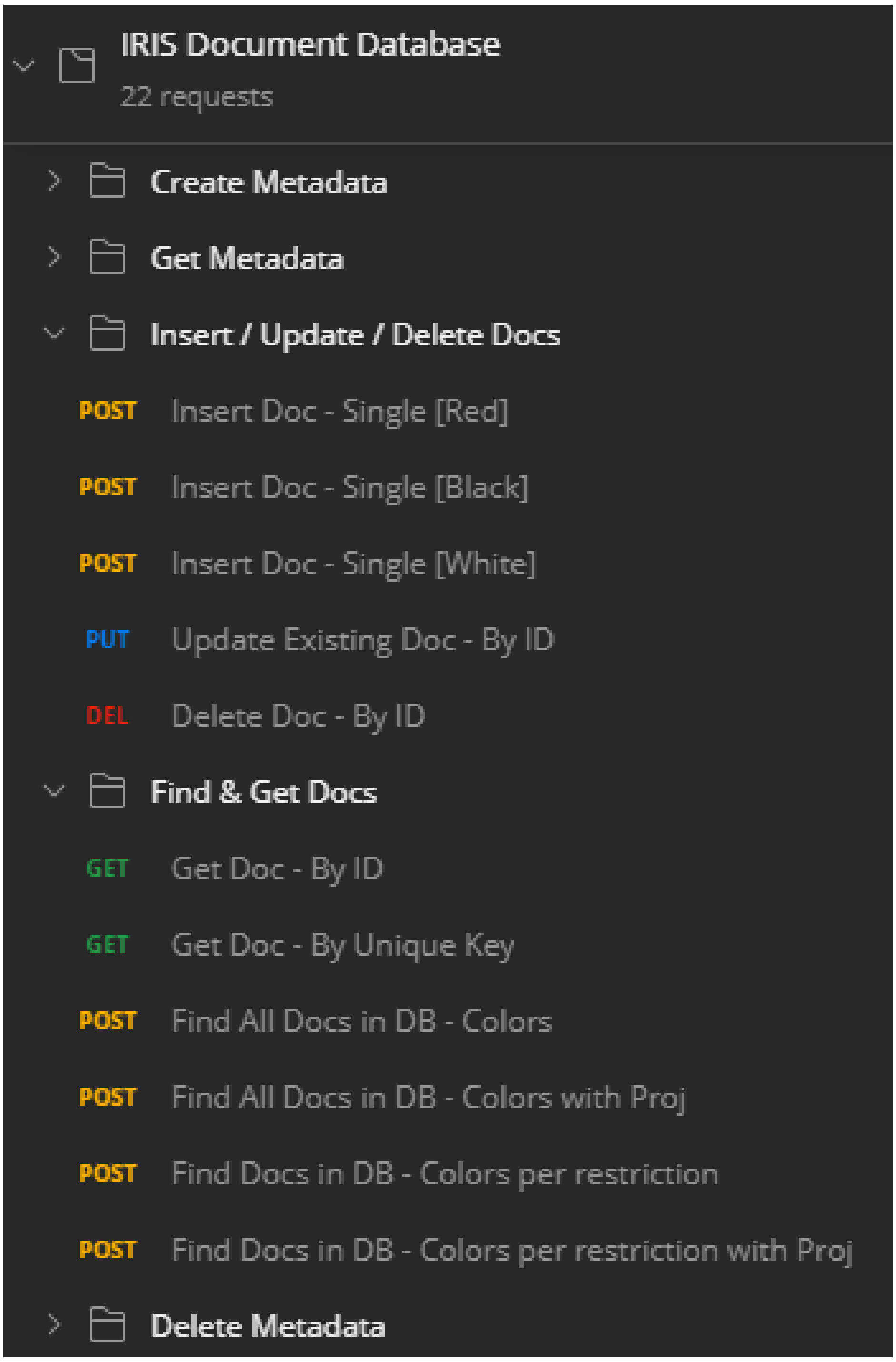
Article
Tani Frankel · Jan 19, 2021 2m read

Open Exchange

Document Database (DocDB) - Sample REST API Calls - Postman Collection

For the benefit of those who want to use the Document Database (DocDB) capabilities within InterSystems IRIS, and specifically the REST API it provides, I put together a Postman Collection that provides samples for several basic calls.

For example:


IRIS Document Database
22 requests

- Create Metadata
- Get Metadata
- Insert / Update / Delete Docs
  - POST Insert Doc - Single [Red]
  - POST Insert Doc - Single [Black]
  - POST Insert Doc - Single [White]
  - PUT Update Existing Doc - By ID
  - DEL Delete Doc - By ID
- Find & Get Docs
  - GET Get Doc - By ID
  - GET Get Doc - By Unique Key
  - POST Find All Docs in DB - Colors
  - POST Find All Docs in DB - Colors with Proj
  - POST Find Docs in DB - Colors per restriction
  - POST Find Docs in DB - Colors per restriction with Proj
- Delete Metadata

The example uses "Color" documents, e.g. Red, Blue, etc, using a sample JSON structure from here.

The Collection includes calls of different "categories" -

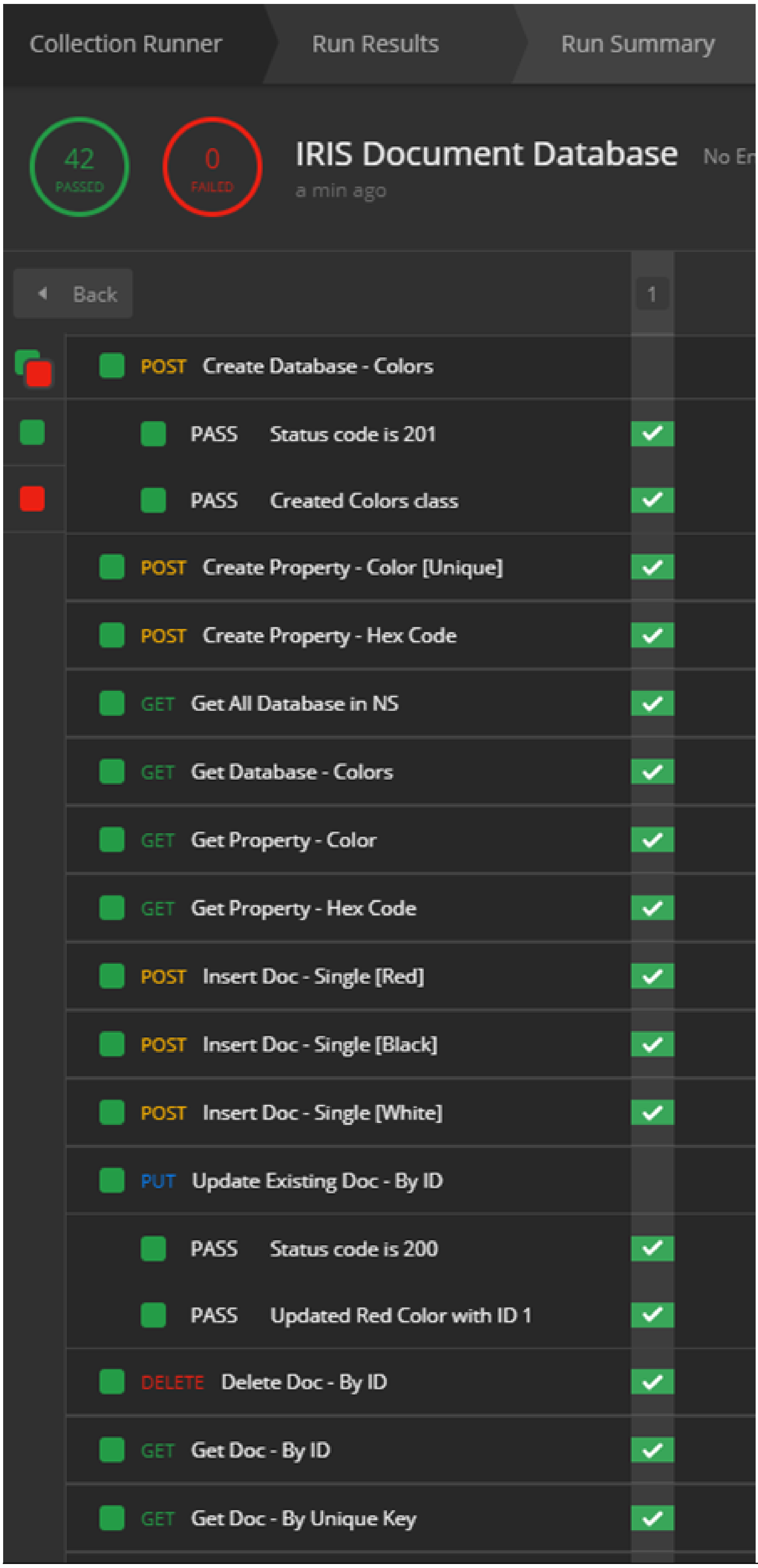
- Create Metadata - create the database and related properties
- Get Metadata - understand what databases and properties are defined
- CUD - create/update/delete of Documents
- Find & Get Documents - retrieve documents according to ID or certain values or criteria
- Delete Metadata - delete properties or databases

The order in which the requests are in the Collection have some internal logic (e.g. first create the database and properties, then insert some data, then retrieve it), but of course you can use it in any order or changes you like.

The order also works well if running the Postman Collection Runner. I added some basic test scripts that allow

Postman to display Passed or Failed status for each call.

For example:



IRIS Document Database No En
a min ago

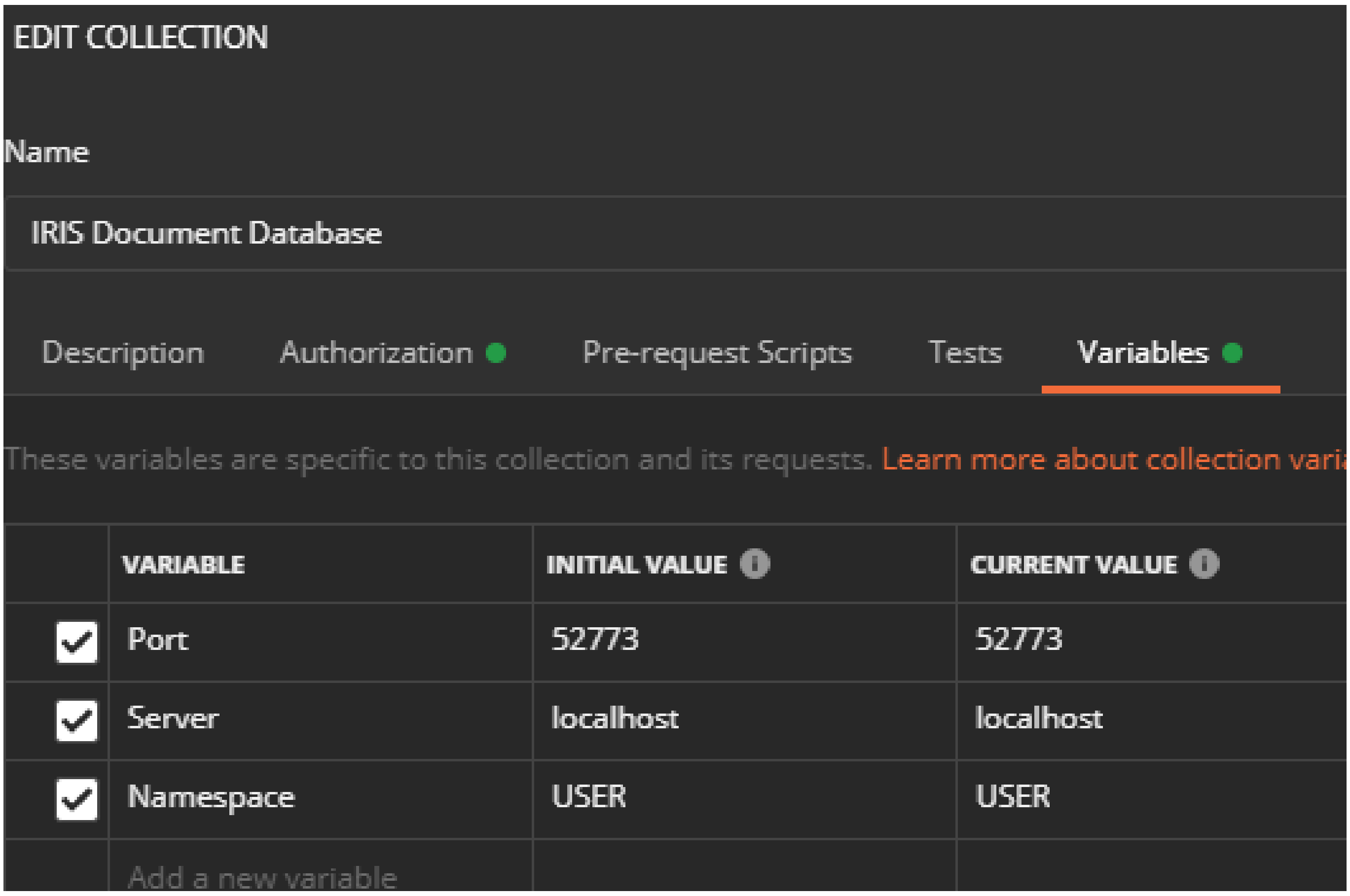
Back 1

```
POST    Create Database - Colors
    PASS    Status code is 201                    ✓
    PASS    Created Colors class                  ✓
POST    Create Property - Color [Unique]          ✓
POST    Create Property - Hex Code                ✓
GET     Get All Database in NS                    ✓
GET     Get Database - Colors                     ✓
GET     Get Property - Color                      ✓
GET     Get Property - Hex Code                   ✓
POST    Insert Doc - Single [Red]                 ✓
POST    Insert Doc - Single [Black]               ✓
POST    Insert Doc - Single [White]               ✓
PUT     Update Existing Doc - By ID
    PASS    Status code is 200                    ✓
    PASS    Updated Red Color with ID 1           ✓
DELETE  Delete Doc - By ID                        ✓
GET     Get Doc - By ID                           ✓
GET     Get Doc - By Unique Key                   ✓
```

Please note the last call **deletes all the document databases within a Namespace** - so **do not run this** unless you really mean to... not manually and not as part of running the whole Collection.

In order to make the calls work on various environments I used Postman's variables feature.

This allows you to change the server name/IP, the port, and the namespace -



EDIT COLLECTION

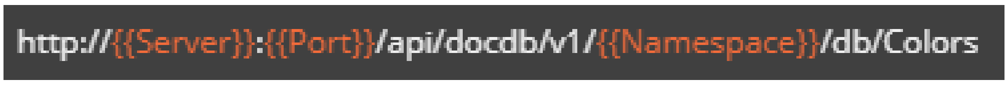
Name

IRIS Document Database

These variables are specific to this collection and its requests. Learn more about collection vari

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	Port	52773	52773
✓	Server	localhost	localhost
✓	Namespace	USER	USER
	Add a new variable		

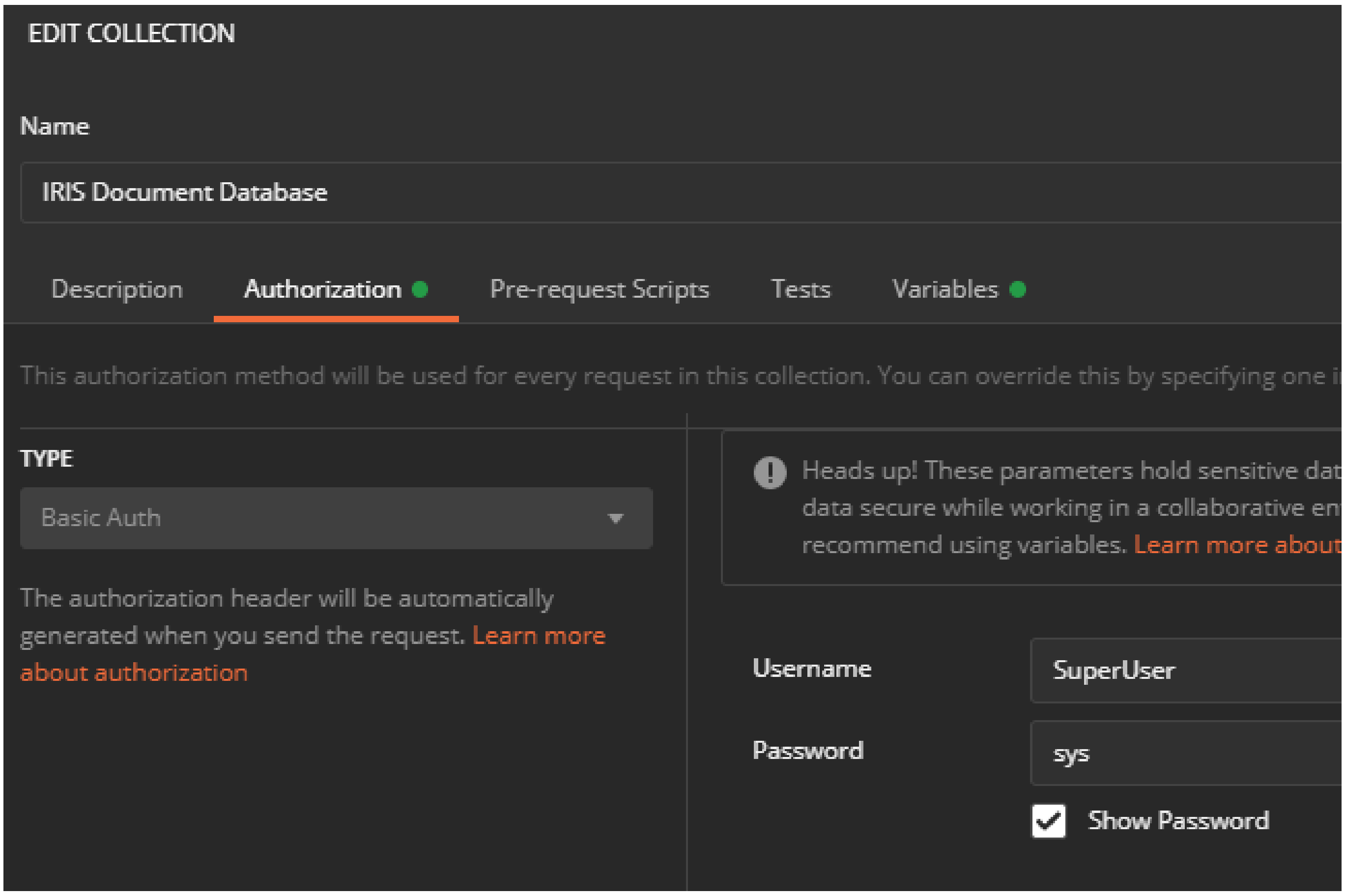
So every call looks something like this:



```
http://{{Server}}:{{Port}}/api/docdb/v1/{{Namespace}}/db/Colors
```

You would also probably need to adapt the authentication part.

I currently have "Basic Authentication" with simply 'SuperUser' and 'sys' -



EDIT COLLECTION

Name

IRIS Document Database

Description | Authorization ● | Pre-request Scripts | Tests | Variables ●

This authorization method will be used for every request in this collection. You can override this by specifying one i

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive dat data secure while working in a collaborative en recommend using variables. Learn more about

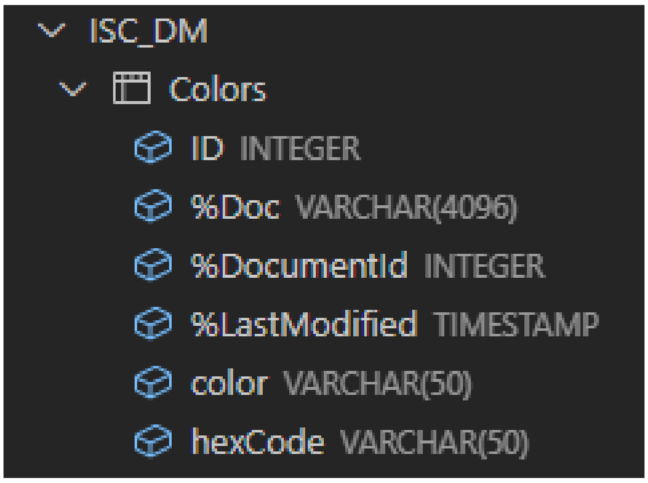
Username: SuperUser

Password: sys

✓ Show Password

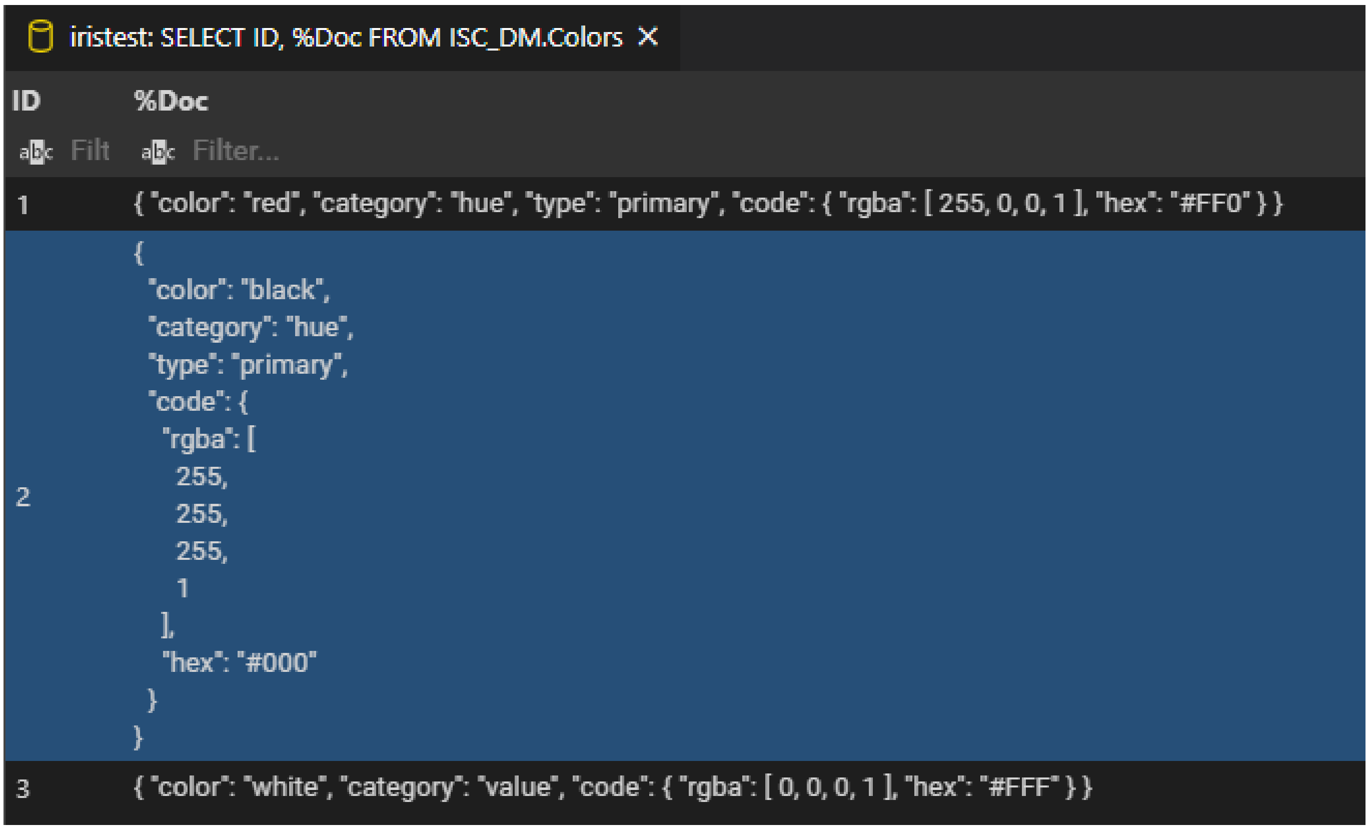
Just to add a ***"Multi-Model" twist*** to this, here's also an example of accessing this data via SQL -

This is the table structure (as viewed in the SQL Tools extension in VSCode):



```
ISC_DM
  Colors
    ID INTEGER
    %Doc VARCHAR(4096)
    %DocumentId INTEGER
    %LastModified TIMESTAMP
    color VARCHAR(50)
    hexCode VARCHAR(50)
```

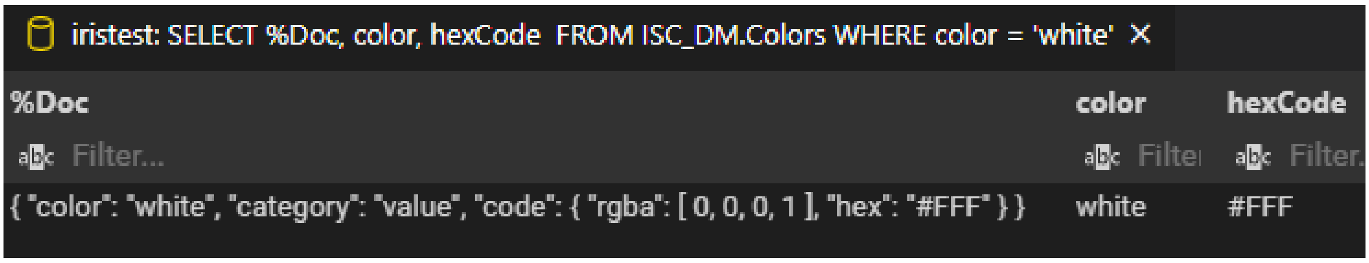
This is a simple SELECT result:



iristest: SELECT ID, %Doc FROM ISC_DM.Colors

ID	%Doc
1	{ "color": "red", "category": "hue", "type": "primary", "code": { "rgba": [255, 0, 0, 1], "hex": "#FF0" } }
2	{ "color": "black", "category": "hue", "type": "primary", "code": { "rgba": [255, 255, 255, 1], "hex": "#000" } }
3	{ "color": "white", "category": "value", "code": { "rgba": [0, 0, 0, 1], "hex": "#FFF" } }

And here's a SELECT with a WHERE clause on one of the properties we defined:



iristest: SELECT %Doc, color, hexCode FROM ISC_DM.Colors WHERE color = 'white'

%Doc	color	hexCode
{ "color": "white", "category": "value", "code": { "rgba": [0, 0, 0, 1], "hex": "#FFF" } }	white	#FFF

#Data Model #Document Data Model (NoSQL) #Multi-model #REST API #InterSystems IRIS #InterSystems IRIS for Health
Check the related application on InterSystems Open Exchange

Source URL:https://community.intersystems.com/post/document-database-docdb-sample-rest-api-calls-postman-collection