


SUPrema

How to get started with BioStar 2 API

Compiled by
George Cartlidge



What is the API or SDK, and what is the difference?

API stands for Application Programming Interface, where SDK stands for Software Development Kit.

An API is a set of functions that allows you to interface with an existing bit of software (I.E BioStar 2), This means any interaction with Suprema devices will need an installation of BioStar 2 to be already installed and that you are actually interacting with the BioStar 2 server and not the devices. Starting with BioStar 2.7.10, there is a new version of the API that includes more functions.

An SDK is a collection of tools that you can use to develop your own applications using a particular framework/platform. Regarding the BioStar 2 SDK, this will be the BS_SDK_V2.dll file. This .DLL contains many APIs that can be used with C++ & C# programming languages. There is also the new G-SDK which functions along the same principal but is based upon gRPC which can support more languages than the existing SDK. Using the SDK, you cut BioStar 2 out completely and directly interact with the device.

Comparing both, the API will be used where you are happy for BioStar 2 to handle most of the work, BioStar 2 would be storing the fingerprint and user data; The API lets you use BioStar 2 from inside another application, so it does not look like BioStar 2. The SDK will be used when you want to connect directly to the device from your own application; Your application would have to handle the connection and the storage of data. The SDK will also allow you to use specific API's that are not available in the BioStar 2 API.

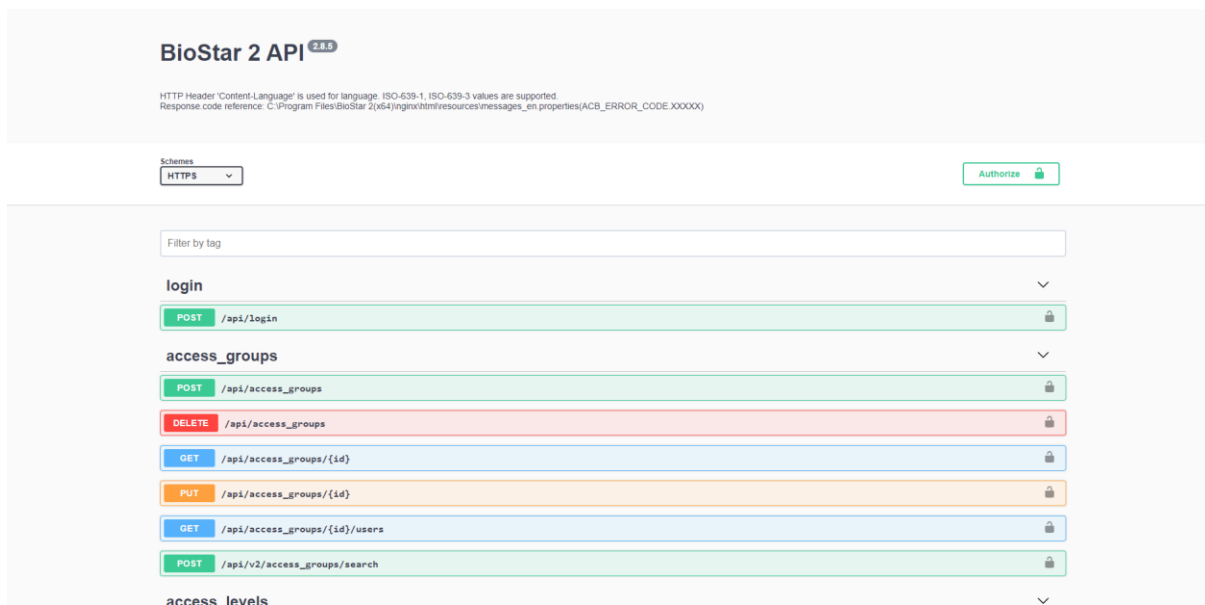
API – Swagger!

Previously, the BioStar 2 API had to be installed separately but starting with BioStar 2.7.10, there is a new API that includes more calls and more features that is packaged together with the install of BioStar 2, so there is no need to install the Local API Server. The local API server is still supported but will no longer be updated. In this document, I will be using the new BioStar 2 API.

On a machine with BioStar 2 installed (Available from <https://www.supremainc.com/en/support/biostar-2-package.asp>)

, The Swagger of the API can be accessed from
'https://*IP_OF_MACHINE*/swagger/index.html'

After navigating to this page, you will see the below screen



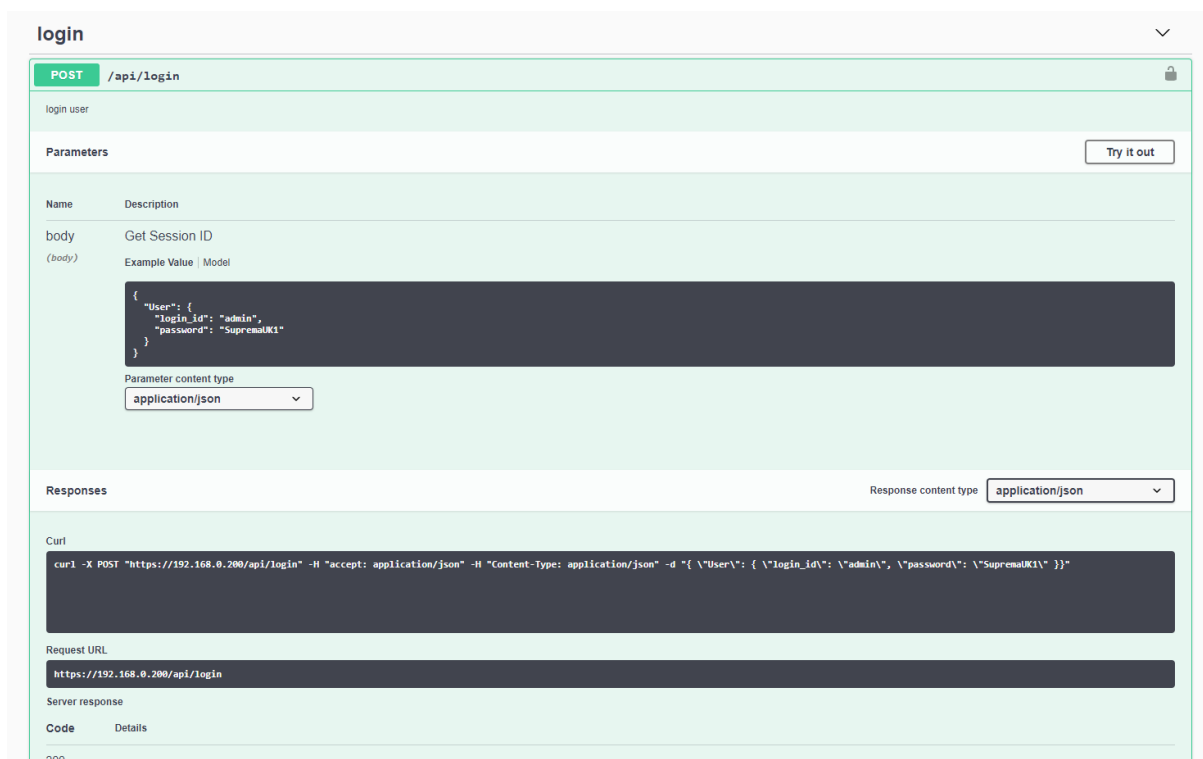
You can see above (And from your screen hopefully) some examples of API calls, you will be able to test some of these by browsing to the IP of your machine followed by the call

(EG. https://*IP OF MACHINE*/api/access_levels/)

Browsing to this page however should bring up 'login required' on the response.

To login to the API, you require a bs-session-id that will then be passed along with the header in all future calls. We can try this ourselves using the Swagger UI and '/api/login'.

NOTE: The username and password will be the admin account given during initial setup (Username should always be 'admin')



The image shows the Swagger UI for the `login` endpoint. The endpoint is `POST /api/login`. The parameters section shows a body parameter named `body` with a description of `Get Session ID`. An example value is provided in a dark box:

```
{  "User": {    "login_id": "admin",    "password": "SupreməUK1"  }}
```

 The parameter content type is set to `application/json`. The responses section shows a response with a status code of 200 and a content type of `application/json`. A curl command is provided in a dark box:

```
curl -X POST "https://192.168.0.200/api/login" -H "accept: application/json" -H "Content-Type: application/json" -d '{"User": {"login_id": "admin", "password": "SupreməUK1"}}'
```

 The request URL is `https://192.168.0.200/api/login`. The server response section shows a status code of 200.

If you click the 'Try it out' button next to parameter, you will be able to enter your own Username and Password (This should be the one used for logging into BioStar 2) and click execute! This just uses the CURL command to poll the API web link. The response body will provide information about the account that you have used to login

login user

Parameters

Cancel

Name	Description
body	Get Session ID

(body) Edit Value Model

```
{
  "User": {
    "login_id": "admin",
    "password": "qwer1234"
  }
}
```

Cancel

Parameter content type
application/json

Execute

Responses

Response content type
application/json

Response body


```
{
  "User": {
    "user_id": "1",
    "name": "Administrator",
    "gender": "1",
    "birthday": "1977-10-08T04:00:00.00Z",
    "photo_exists": "false",
    "pin_exists": "false",
    "login_id": "admin",
    "password_exists": "true",
    "updated_count": "3",
    "last_modified": "42",
    "start_datetime": "2001-01-01T00:00:00.00Z",
    "expiry_datetime": "2030-12-31T23:59:00.00Z",
    "security_level": "0",
    "display_duration": "20",
    "display_count": "3",
    "permission": {
      "id": "1",
      "name": "Administrator",
      "description": "this is a permission for Administrator",
      "filter": {
        "UserGroup": [
          "1"
        ],
        "DeviceGroup": [
          "1"
        ]
      }
    }
  }
}
```

Where the response header will provide you with the bs-session-id

Response headers

```
bs-session-id: da91612a63ba41daa9512f819b4bd071
cache-control: no-cache, no-store, max-age=0, must-revalidate
content-length: 1960
content-type: application/json; charset=UTF-8
date: Wed, 16 Sep 2020 10:52:02 GMT
expires: 0
pragma: no-cache
status: 200 OK
strict-transport-security: max-age=31536000 ; includeSubDomains
x-content-type-options: nosniff
x-frame-options: SAMEORIGIN
x-xss-protection: 1
```

If you take a copy of this bs-session-id, you will be able to input it at

the top of the page under the  button.

Available authorizations x

bs-session-id (apiKey)

Name: bs-session-id

In: header

Value:

Authorize Close

Once this has been inputted, you will be able to use the rest of the API calls, as the swagger will automatically pass this bs-session-id along when posting to the APIs.

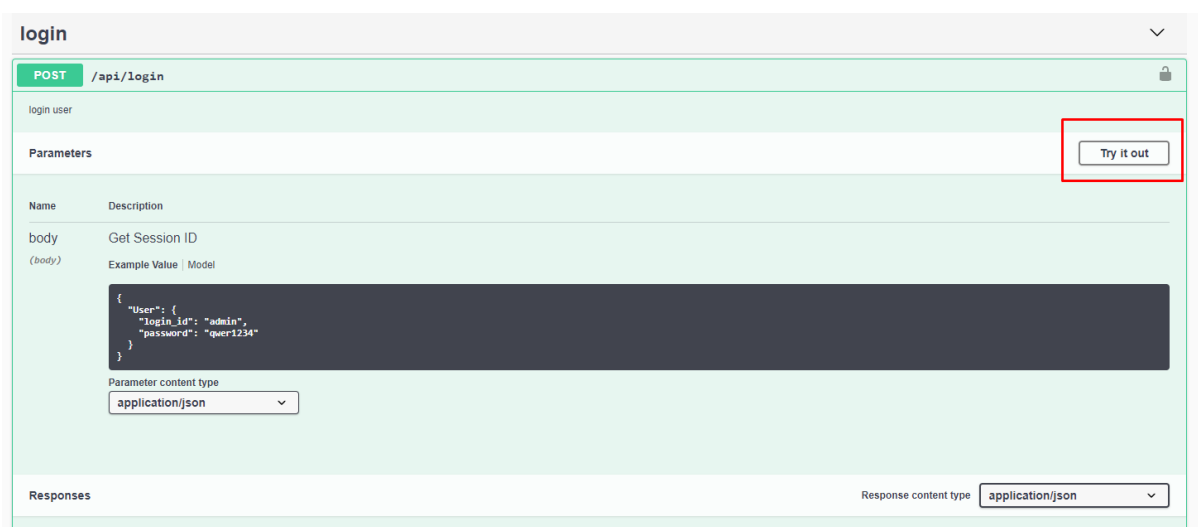
API - Using cURL

Our API swagger is a good view of how you would go about interacting with the BioStar 2 server, you could even do something very simple like interacting with BioStar 2 with CMD (I added a little bit of PowerShell to hide the password) and cURL commands!

Another option would be Postman, available at <https://www.postman.com/>, this does use curl commands, but provides a useable interface for sending and receiving responses. For ease of use (To limit the amount of installs) I will be using cURL directly.

cURL is simply a software project that provides a way of transferring data (Downloading and uploading) through network protocols, an even better point is that it comes pre-packaged with windows, so you don't need to download anything to use it!

When using the API swagger, clicking try it out next to any calls (Like above) will let you edit the parameters.



login user

Parameters Cancel

Name	Description
body (body)	Get Session ID Edit Value Model

```
{
  "User": {
    "login_id": "admin",
    "password": "quer1234"
  }
}
```

Cancel

Parameter content type
application/json

Execute

Responses Response content type application/json

You can then click 'Execute', this will send the command to the server using cURL, you can see and take a note of the curl command below the execute button.

Responses Response content type application/json

Curl

```
curl -X POST "https://192.168.0.200/api/login" -H "accept: application/json" -H "Content-Type: application/json" -d "{"User": {"login_id": "admin", "password": "quer1234"}}"
```

Request URL

```
https://192.168.0.200/api/login
```

Server response

Code	Details
------	---------

Now we have a curl command, you can input this through CMD to interact with the server, which should give the same response as the swagger.

```
Microsoft Windows [Version 10.0.19042.630]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\gcartledge>curl -X POST "https://192.168.0.200/api/login" -H "accept: application/json" -H "Content-Type: application/json" -d "{"User": {"login_id": "admin", "password": "quer1234"}}"
```

```
{
  "code": "101",
  "link": "https://support.supremainc.com/en/support/home",
  "message": "Failed to login for invalid username or password"
}
```

In the next steps, you'll see how we can do this using only CMD.

API – Login!

The below **batch** script is an example of the login API using the curl post generated by the swagger, where you can enter the IP of the server (It is worth noting, to use the API with https, you will need the SSL certificate installed on the machine), enter a username/password and it logs you in with a curl output response that contains a valid bs-session-id, the end of the script then outputs that line to a separate .txt file that will be used in the next script!

```
@echo off

:: Prompts user for IP of the server, Sets variable then exports to a .txt file

echo Please enter IP of server:

set /P IP=

del IP.txt 2>NUL

@echo %IP%> IP.txt

echo.

::Prompts user for Username input and sets as variable

echo Please enter Username:

set /P UserID=

echo.

::Prompts user for password input but hides the characters using PowerShell, then sets as variable

powershell -Command $pword = read-host "Enter password" -AsSecureString ; ^
$BSTR=[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($pword) ; ^
[System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR) > .tmp.txt

set /p UserPW=<.tmp.txt & del .tmp.txt

::Uses cURL to sent the command to server, using variables inputted above, exports to curloutput.txt

curl -X POST "https://%IP%/api/login" -H "accept: application/json" -H "Content-Type: application/json"
-d "{\"User\": { \"login_id\": \"%UserID%\", \"password\": \"%UserPW%\" }}" -o curloutput.txt -i

::Searches curloutput.txt for the BS-Session-ID Line and exports it singularly to a .txt file.

del BSSession.txt 2>NUL

findstr /b /r "bs-session-id" curloutput.txt >>BSSession.txt
```


API – Add User!

The below **batch** script is again very simplistic but allows you to input a user into BioStar 2 using the curl command generated by the swagger. It first uses the BSSession.txt generated by the Login script previously used and sets this as a variable, you then just need to enter the ID, name and email of the new user.

```
@echo off

:: Uses files previously generated using API Login.cmd to set IP and BS-Session-ID

set /P IPAddress= <IP.txt
set /P bsession= <BSSession.txt

::Prompts for userID of user to be added
echo Please enter New User ID:
set /P ID=
echo.

::Prompts for name of User to be added
echo Please enter Name:
set /P Name=
echo.

::Prompts for email of user to be added
echo Please enter Email:
set /P Email=
echo.

::Puts all of the above through a curl command
curl -X POST "https://%IPAddress%/api/users" -H "accept: application/json" -H "%bsession%" -H
"Content-Type: application/json" -d "{\"User\": { \"name\": \"%name%\", \"email\": \"%email%\",
\"user_id\": \"%ID%\", \"user_group_id\": { \"id\": \"1\" }, \"disabled\": \"false\",
\"start_datetime\": \"2001-01-01T00:00:00.00Z\", \"expiry_datetime\": \"2030-12-
31T23:59:00.00Z\", \"cards\": [] } }"
```

This is setting the basics of the user, as within this, I have set 'User group ID' to always be '1', as well the start and expiry to the default BioStar 2 values (2001 – 2030).