

# Getting started with the REST API

There's no better way to automate Syncplify Server! management and configuration tasks than by scripting against our comprehensive REST API.

This article assumes basic familiarity with scripting concepts (bash, zsh, fish, PowerShell, or languages like Python). No advanced expertise is required – just the ability to execute API calls and handle responses.

We've invested significant effort in creating detailed OpenAPI v3 documentation for every REST endpoint. This specification is:

- Available [in the official manual](#)
- Accessible through [web-based documentation UIs](#)
- Downloadable as raw JSON files for integration with API tools like Postman, Insomnia, or Bruno

**Key Security Consideration:** As enterprise security software, Syncplify Server! intentionally avoids long-lived API keys. Instead, our authentication flow requires:

1. Initial login API call to obtain a time-limited JWT
2. Use of this session token for subsequent API calls
3. Automatic token expiration for enhanced security

Below is a practical **bash** example (requires `curl` and `jq`, assumes local Syncplify Server! installation at `127.0.0.1:6443`):

```
#!/bin/bash

# Fetch the JSON response to the /login API and extract the token
token=$(curl --request GET --url https://127.0.0.1:6443/api/v1/sa/login --header 'authorization: Basic XXXXXXXX' | jq -r '.token')

# Call any other REST API using the authorization token you received upon login

# Example: list all virtual sites
```

```

curl --request GET \
  --url https://127.0.0.1:6443/api/v1/sa/vsites \
  --header "Authorization: Bearer $token"

# Example: modify some aspects of the SuperAdmin global configuration
curl --request PATCH \
  --url https://127.0.0.1:6443/api/v1/sa/globalconfig \
  --header "Aauthorization: Bearer $token" \
  --header 'Content-Type: application/json' \
  --data '{
    "jwtLifeSpan": 30,
    "metricsAllowList": [
      "192.168.10.0/32"
    ]
  }'

#At the end of your session, do not forget to logout
curl --request GET \
  --url https://127.0.0.1:6443/api/v1/sa/logout \
  --header "Authorization: Bearer $token"

```

The script here above:

1. Performs the login as SuperAdmin, receiving a short-lived JWT (token) that authorizes only SuperAdmin-related API calls
2. Calls an API (GET) to receive the list and details of all virtual sites
3. Calls another API (PATCH) to modify a few minor aspects of the SuperAdmin UI configuration
4. Calls the logout API to invalidate the temporary JWT, from this point on every API call done using this JWT (even valid ones) will be rejected as unauthorized

Here's a **PowerShell** version of the exact same script (with the exact same assumptions as above):

```

# Login and get token (note case-insensitive headers in PowerShell)
$loginResponse = Invoke-RestMethod -Uri "https://127.0.0.1:6443/api/v1/sa/login" `
  -Method Get `
  -Headers @{ 'authorization' = 'Basic XXXXXXXXX' }

$token = $loginResponse.token

# List all virtual sites

```

```

$vSites = Invoke-RestMethod -Uri "https://127.0.0.1:6443/api/v1/sa/vsites" `
    -Method Get `
    -Headers @{ Authorization = "Bearer $token" }

$vSites | Format-Table # Display results in table format

# Modify global configuration (note typo in original header name)
$body = @{
    jwtLifeSpan = 30
    metricsAllowList = @('192.168.10.0/32')
} | ConvertTo-Json

Invoke-RestMethod -Uri "https://127.0.0.1:6443/api/v1/sa/globalconfig" `
    -Method Patch `
    -Headers @{
        # Original script had typo: 'Aauthorization' instead of 'Authorization'
        'Authorization' = "Bearer $token"
        'Content-Type' = 'application/json'
    } `
    -Body $body

# Logout to end session
Invoke-RestMethod -Uri "https://127.0.0.1:6443/api/v1/sa/logout" `
    -Method Get `
    -Headers @{ Authorization = "Bearer $token" }

```

If your server is running on a self-signed certificate, you may also want to consider the following:

```

# For self-signed certificates (PowerShell 7+):
$ProgressPreference = 'SilentlyContinue' # Suppress progress noise

# Add -SkipCertificateCheck to ALL Invoke-RestMethod calls
Invoke-RestMethod ... -SkipCertificateCheck

```

**Please note:** all of the example code above shows you how to use REST APIs in the SuperAdmin category, exclusively to perform SuperAdmin actions and tasks. To perform Admin tasks, the logic is exactly the same but you will have to login using the Admin login API to obtain an Admin JWT instead.

This is the Admin login API called from a **bash** script:

```
#!/bin/bash
```

```
# Fetch the JSON response to the Admin API /login endpoint and extract the token
```

```
token=$(curl --request GET --url https://127.0.0.1:6443/api/v1/adm/login --header 'authorization: Basic  
XXXXXXXXXX' | jq -r '.token')
```

And its **PowerShell** equivalent:

```
# Admin login API, and acquisition of the Admin JWT token from the response
```

```
$loginResponse = Invoke-RestMethod -Uri "https://127.0.0.1:6443/api/v1/adm/login" `
```

```
-Method Get `
```

```
-Headers @{ 'authorization' = 'Basic XXXXXXXXX' }
```

```
$token = $loginResponse.token
```

We hope that this brief introduction will help you get started using our REST API more quickly and painlessly.

---

Revision #3

Created 26 March 2025 20:38:29 by DevTeam

Updated 7 April 2025 23:34:35 by DevTeam