digital nirvana™

Monitor v8.0

(Broadcast Compliance and Monitoring Solution)

API Documentation

Version History

<Stripped>	**<Stripped>**
<Stripped>	**<Stripped>**
<Stripped>	**<Stripped>**
<Stripped>	**<Stripped>**
<Stripped>	**<Stripped>**

Table of Contents

<----------TOC – Stripped---------->

Introduction

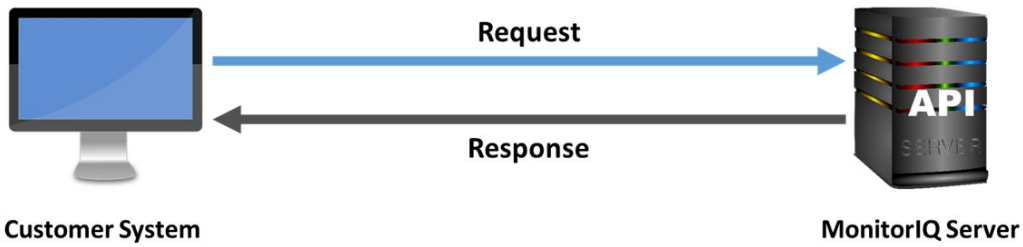
The MonitorIQ APIs offer seamless integration with diverse workflows, enabling the creation of tailored interfaces that align with specific business needs. Following machine setup, system administrators can configure channels through the MonitorIQ front end. Once channels are configured, API calls to the machine facilitate a range of tasks, including schedule management, recording searches, content repurposing through clip creation, categorization, and uploading to various FTP locations.

API Overview

Application programming interface (API) is a code that acts as an interface between two applications. Using APIs, the applications can communicate with each other. It defines a set of protocols to request services from an operating system (OS) or application and expose data within different contexts and across multiple channels. Different types of data are shared using the API request/response process.

How API Works?

An API acts as a medium between a client and a server. When a customer wants to access the services from a server, he/she sends a request (API call) to that server. If the customer has sent a proper request, then the server responds to that request and sends the required services back to the customer. Here, the API passes customer request to the server and gives back the response from the server. The working functionality of API is shown in the figure below:



MonitorIQ API Calls Description

The following are the different types of API calls to the MonitorIQ server.

- **API Call For Managing Channels**

  The *<stripped>* endpoint in the API calls is used to manage the channels in MonitorIQ like creating a new channel, viewing all the channels/a specific channel, updating and/or deleting existing channels, etc.

- **API Calls For Managing Schedules**

  The *<stripped>* endpoint in the API calls is used to manage the channels' recording schedules like adding a new schedule, viewing and/or deleting existing schedules, etc.

- **API Calls For Managing Recording Channels**

  The *<stripped>* endpoint in the API calls is used to manage the channels' recordings like viewing a specific recording/all the recordings, deleting an existing recording, and stop an ongoing recording.

- **API Calls For Managing Cut Clips**

  The *<stripped>* endpoint in the API calls is used to manage the clips generated from recordings like creating a new clip, viewing a specific/all the clips, deleting an existing clip, and cancelling a clipping process.

- **API Calls For Managing Bookmarks**

  The *<stripped>* endpoint in the API calls is used to manage the bookmarks like creating a new bookmark, viewing a specific bookmark/all the bookmarks, and deleting an existing bookmark.

- **API Calls For Managing Cut Clip Playlists**

  The *<stripped>* endpoint in the API calls is used to manage the cut clip playlists like creating a new playlist, viewing a specific playlist/all the playlists, and deleting an existing playlist.

- **API Calls For Managing System Settings**

  The *<stripped>* and *<stripped>* endpoints in the API calls are used to manage the system settings like viewing a specific FTP profile, all FTP profiles, and retrieving frames from recordings.

- **API Calls For Monitoring Management**

  The *<stripped>* and *<stripped>* endpoints in the API calls are used to retrieve alerts and live recording frames.

API Requests

The MonitorIQ APIs are built on RESTful architecture. All requests should send through HTTPS only. API endpoints share the same URL and can perform different actions based on the HTTP method used to access it. Use appropriate HTTP method to perform an intended action (For accepted HTTP methods refer to the HTTP Methods section below). An HTTP status code is returned with each request that indicates the status of that request (For detailed error codes used to refer to the Status Codes section). All the responses to the API calls (endpoints) are in JSON format.

HTTP Methods

The HTTP methods are used to indicate the desired action to be performed for a given resource. As per REST guidelines, a specific HTTP method is used for a specific call to the server. The following are common HTTP methods:

Method	Description
GET	The GET method is used to get data from an API endpoint. No data will be modified using GET requests.
POST	The POST method is used to create a new item. The request parameters should be passed as JSON format. The response body will be the newly created item.
PUT	The PUT method is used to update an existing item. The request parameters should be passed as JSON format. The response body will be the updated item.
DELETE	The DELETE method is used to delete an item. The response will return empty if successful.

There are some more HTTP methods available such as HEAD, CONNECT, TRACE, OPTIONS, and PATCH.

API Responses

All the API responses are obtained in JSON format.

HTTP Status Codes

The HTTP status codes are part of a response from the server. This indicates whether an HTTP request has been successfully completed or failed due to an error. These responses are classified into five different groups: informational responses, successful responses, redirects, client errors, and server errors. The following are some of the common status codes:

HTTP Code	Description
200	**OK** - Returns this code on successful request.
201	**Created** - Returns this code after successful creation of a new item.
204	**No Content** - Returns this code after Successful DELETE request.

400	**Bad Request** - Returns this code if there is an issue with the request parameters or request body.
401	**Unauthorized** - Returns this code if the token is invalid or no access to the requested endpoint.
403	**Forbidden** - Returns this code if there is no access to the requested endpoint.
404	**Not Found** - Returns this code if the requested endpoint does not exist.
409	**Conflict** - Returns this code if the requested action conflicts with another ongoing request.
500	**Internal Server Error** - Returns this code if there is a problem with the service from MonitorIQ Side.
503	**Service Unavailable** - Returns this code if the service is not available.

Errors

If a request's parameters contain errors, then the response obtained for that request includes a detailed error code and message. The error codes are returned in the JSON format are different from the HTTP status codes. An HTTP status code 400 is returned if there are any errors in the request parameters.

Sample Error Response

The following are some of the errors that might be obtained for an improper/ bad request. When a customer sends a request to add an existing channel, the following error message could be obtained:

<---------- *stripped*---------- >

Login

Description

Use this API call to login to the application to access its services. This API call requires *<stripped>* endpoint at the end of the URL.

URL

http://<ip-or-url>/*<stripped>*/*<stripped>*

Request Method

POST

Input Parameters

- name: string (Provide the username as the value to this parameter)
- password: string (Provide the password as the value to this parameter)

Request Body

```
POST /<stripped>/<stripped> HTTP/1.1
Host: http://<ip-or-url>/
Connection: keep-alive
Content-Length: 246
Accept: application/json, text/plain, */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93
Safari/537.36
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryreHSQwRfiPewOR1b
Origin: http://<ip-or-url>/
Referer: http://<ip-or-url>/<stripped>/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: ogbgcUv2iv5hJe42Jf85Pv45MuynfvGGQ2=eyJpdiI6ImlN;
laravel_session=eyJpdiI6Iitic1hMUi9IVGJOay9JUlcr;
d6oU3s0yD5EVM9TgWUMG86m78GR2cd3Gs
```

Output Response


```
{
 "data":{"id":1,
         "name":"bob",
         "email":"support@example.com",
         "first_name":null,
         "last_name":null,
         "status":"ACTIVE",
         "auth_type":"mmp_pwd",
         "objectguid":null,
         "api_token":"464|CzKbbSAJMoEoHSZx9",
         "remote_api_token":"41c4aff81c405a4",
         "enable_api_access":"YES",
         "permissions":["VIEW_ALERTS",
                        "ADD_SCHEDULES",
                        "DELETE_SCHEDULES",
                        "MANAGE_ENCODING_SETTINGS",
                        "SEARCH_RECORDINGS",
                        "VIEW_SCHEDULES",
                        "MANAGE_USERS",
                        "MANAGE_ROLES",
                        "MANAGE_TIMEZONE_NTP_SETTINGS",
                              .
                              .
                              .
  "bearer_token":"498|LnihKocoxx8XEYseuYt4fe10t1RP ",
         "timezone":"Asia\/Kolkata"
          }
  }
```


HTTP Status Codes

- 200 OK
- 401 Unauthorized
- 404 Not Found

Channels

Description

Use this API call to bring live channels in different pages like Recordings, Viewer, etc. This API call requires <*stripped*> endpoint and “record_channels” input parameter in the URL.

URL

http://<ip-or-url>/mmp-api/<*stripped*>?record_channels=true

Request Method

GET

Request Parameters

- record_channels: true | false

Request Header

Pass the access token obtained from response of Login API in the **Authorization** header and use “**Bearer**” keyword before the token as shown below.

Authorization: Bearer xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Request Body

GET /mmp-api/<*stripped*>?record_channels=true HTTP/1.1

Host: http://<ip-or-url>/

Connection: keep-alive

Accept: application/json, text/plain, */*

Authorization: Bearer 498|LnihKocoxx8XEYseuYt4fe10t1RP

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/xx.x.xxxx.xxxSafari/537.36

Content-Type: application/json

Referer: http://<ip-or-url>/<*stripped*>/

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.9

Cookie: ogbgcUv2iv5hJe42Jf85Pv45QRetzdMuynfvG;
6mvDy9POuhKWSyF0zCpwg0DXB5KUSkPh7PKsD;
laravel_session=eyJpdiI6IlNaM3JFdGFtSnpLOGdsRlV;
fkLKInHgc2bs7aAGa06mIWPJSUBsCFJk9S2xb

Response Parameters

None

Response Body


```
{
   "data": [
      {
         "chan_id": 81757,
         "callsign": "TV-102",
         "visible": 1,
         "channel_logo": " http://<ip-or-url>/miq/<stripped>/img/novideo.png",
         "channel_type": "tv",
         "source_name": "HDMI-VS2",
         "m3u8UrlPath": "",
         "isLive": true,
         "rec_exists": "0",
         "programStartDateTime": "",
         "group_name": [],
         "hostname": "miqlab"
      },
      .
      .
      .
   ],
   "status": 200,
   "message": "Successful",
   "total": 8
```

}

HTTP Status Codes

- 200 OK
- 401 Unauthorized
- 404 Not Found