Visual Array REST API Documentation

> This document describes the REST API for Visual Array version 1.3.0 (VA 1.3.0). Ensure your deployment matches VA 1.3.0 to avoid discrepancies in endpoints and parameters.

Introduction

The Visual Array REST API enables developers to manage AV matrix routing, video wall control, and resource management for PlexusAV's Visual Array system. This API is designed for system integrators, developers, and administrators who need programmatic control over AV equipment. This document provides detailed information about endpoints, authentication, request/response formats, and best practices.

Target Audience: Developers with experience in REST APIs, JSON, and AV system integration.

How to Use This Document:

- Start with the Authentication section to understand session management.
- Use the API Discovery endpoint to explore available features.
- Refer to specific sections (e.g., Resource Management) for endpoint details.
- Check Examples for cURL commands and code snippets.

Table of Contents

Base URL

All API endpoints are prefixed with `/api/v1/`. For example, the login endpoint is accessible at `https://<your-server>/api/v1/auth/login`.

Security

The API implements multiple security measures to protect sensitive data and prevent unauthorized access:

Transport Security

- **HTTPS Required**: All production deployments must use HTTPS
- **Session Management**: Secure session tokens with automatic expiration
- **Rate Limiting**: Prevents brute force and abuse attacks
- **URL Decoding**: Automatic decoding of URL-encoded special characters in query parameters, path parameters, and request bodies

Security Best Practices

1. **Use HTTPS**: Always use HTTPS for secure password transmission
2. **HTTPS Only**: Never transmit credentials over HTTP
3. **Secret Rotation**: Regularly rotate API secrets and keys
4. **Session Validation**: Always validate session tokens before requests
5. **Error Handling**: Don't expose sensitive information in error messages
6. **URL Encoding**: The API automatically handles URL-encoded special characters, but it's still recommended to properly encode parameters for compatibility

See URL Encoding Guidelines for the complete rules, examples, and recommended language helpers.

URL Encoding Guidelines

The API automatically handles URL-encoded special characters, but proper encoding is recommended for maximum compatibility across different systems and tools.

URL Encoding Standards

RFC 3986 Compliance: Use standard percent-encoding as defined in RFC 3986 for all URL parameters.

Character Encoding Rules:

- **Reserved Characters**: Always encode `&`, `=`, `?`, `#`, `%`, `+`, `/`, `:`, `@`, `!`, `$`, `'`, `(`, `)`, `*`, `,`, `;`
- **Unreserved Characters**: Keep `A-Z`, `a-z`, `0-9`, `-`, `_`, `.`, `~` as-is
- **Spaces**: Encode as `%20` (not `+`)
- **Unicode**: Encode non-ASCII characters using UTF-8 followed by percent-encoding

Implementation Examples

JavaScript:

```
// Encode individual parameters
const encodedName = encodeURIComponent("My Device & Name");
const encodedType = encodeURIComponent("Video/Audio");

// Build URL with encoded parameters
const url = `/api/v1/resource/resources?name=${encodedName}&type=${encodedType}`;
```

Python:

```
import urllib.parse

# Encode individual parameters
encoded_name = urllib.parse.quote("My Device & Name")
encoded_type = urllib.parse.quote("Video/Audio")

# Build URL with encoded parameters
url = f"/api/v1/resource/resources?name={encoded_name}&type={encoded_type}"
```

Java:

```
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;

// Encode individual parameters
String encodedName = URLEncoder.encode("My Device & Name",
StandardCharsets.UTF_8);
String encodedType = URLEncoder.encode("Video/Audio", StandardCharsets.UTF_8);

// Build URL with encoded parameters
String url = "/api/v1/resource/resources?name=" + encodedName + "&type=" +
encodedType;
```

C#:

```
using System;

// Encode individual parameters
string encodedName = Uri.EscapeDataString("My Device & Name");
string encodedType = Uri.EscapeDataString("Video/Audio");

// Build URL with encoded parameters
string url = $"/api/v1/resource/resources?name={encodedName}&type={encodedType}";
```

PHP:

```
// Encode individual parameters
$encodedName = urlencode("My Device & Name");
$encodedType = urlencode("Video/Audio");

// Build URL with encoded parameters
$url = "/api/v1/resource/resources?name=" . $encodedName . "&type=" .
$encodedType;
```

Common Character Encodings

Character	Encoding	Example
Space	`%20`	`My Device` → `My%20Device`
Ampersand	`%26`	`Video&Audio` → `Video%26Audio`
Plus	`%2B`	`HD+4K` → `HD%2B4K`
Forward Slash	`%2F`	`Video/Audio` → `Video%2FAudio`
Question Mark	`%3F`	`Help?` → `Help%3F`
Hash	`%23`	`Section#1` → `Section%231`
Percent	`%25`	`50%` → `50%25`
Equals	`%3D`	`key=value` → `key%3Dvalue`

URL Encoding Best Practices

1. **Always Encode**: Encode all parameter values, even if they appear to contain only safe characters
2. **Use Built-in Functions**: Leverage your programming language's built-in URL encoding functions
3. **Test Edge Cases**: Test with special characters, spaces, and Unicode characters
4. **Consistent Encoding**: Use the same encoding method across your entire application
5. **Documentation**: Document your encoding approach for team consistency

Authentication

Authentication is required for most endpoints. The API uses session-based authentication, where a session ID is obtained via the login endpoint and included in subsequent requests as a cookie.

Important Note: This API implementation primarily supports cookie-based authentication using `auth_session=<session-id>`. While the documentation may mention Bearer token authentication, the current implementation may not fully support this method. It is recommended to use cookie authentication for all API requests.

Login

Purpose: Authenticate users and obtain a session ID.

Endpoint: `POST /api/v1/auth/login`

Description: Authenticates a user with a username and password, returning a session ID. The session ID is set as a cookie (`auth_session`) and can also be used in the `Authorization` header.

Security Features:

- **HTTPS Required**: All authentication requests must use HTTPS
- **Session Management**: Secure session-based authentication

Request Body:

```
{
  "user": "admin",
  "password": "proav101"
}
```

Parameters:

Parameter	Type	Required	Description
`user`	String	Yes	Username for authentication
`password`	String	Yes	Password

Response (200 OK):

```
{
  "success": true,
  "data": {
    "sessionId": "uuid-session-id",
    "cookie": {
      "key": "auth_session",
      "value": "uuid-session-id"
    }
  },
  "message": "Success",
```

```
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`sessionId`	String	Unique session identifier for authentication
`cookie`	Object	Cookie information for session management
`cookie.key`	String	Cookie name (`auth_session`)
`cookie.value`	String	Session ID value

Error Responses (401 Unauthorized):

Invalid Credentials:

```
{
  "success": false,
  "data": null,
  "message": "Invalid username or password",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Usage: Include the session ID in subsequent requests:

- **Cookie**: `auth_session=uuid-session-id`
- **Header**: `Authorization: Bearer uuid-session-id` (Note: This method may not be supported in all implementations)

Security Implementation:

Code Example:

```
// Send request with plaintext password
const response = await fetch('/api/v1/auth/login', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    user: username,
    password: password  // Password
  })
});
```

Logout

Purpose: Terminate the current user session.

Endpoint: `POST /api/v1/auth/logout`

Description: Invalidates the session and clears the `auth_session` cookie. This endpoint can be called with or without a session ID in the request body. If no session ID is provided, it will only clear the cookie.

Request Body:

```
{
  "sessionValue": "uuid-session-id"
}
```

Parameters:

Parameter	Type	Required	Description
`sessionValue`	String	No	Session ID to invalidate (optional)

Response (200 OK):

```
{
  "success": true,
  "data": {},
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data`	Object	Empty object (no additional data)

Error Response (500 Internal Server Error):

```
{
  "success": false,
  "data": null,
  "message": "Internal server error",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Usage:

- **With Session ID**: Include `sessionValue` in request body and use `Cookie: auth_session=<session-id>` header
- **Without Session ID**: Use `Cookie: auth_session=<session-id>` header (cookie will be cleared automatically)

Security Notes:

- **Session Invalidation**: When a session ID is provided, the backend will invalidate the session on the server
- **Cookie Clearing**: The `auth_session` cookie is automatically cleared regardless of the request method
- **HTTPS Required**: All logout requests must use HTTPS in production
- **No Re-authentication**: After logout, users must re-authenticate to access protected endpoints

Validate Session

Purpose: Check if a session is active.

Endpoint: `POST /api/v1/auth/validate`

Description: Verifies the validity of a session ID.

Request Body:

```
{
  "sessionValue": "uuid-session-id"
}
```

Parameters:

Parameter	Type	Required	Description
`sessionValue`	String	Yes	Session ID to validate

Response (200 OK):

```
{
  "success": true,
  "data": {
    "valid": true
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`valid`	Boolean	Indicates if the session is valid

Error Response (200 OK):

```
{
  "success": true,
  "data": {
    "valid": false
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Error Response Fields:

Field	Type	Description
`valid`	Boolean	Always `false` for invalid sessions

API Discovery

Get API Information

Purpose: Retrieve metadata about the API.

Endpoint: `GET /api`

Description: Returns details about the API, including version, endpoints, and supported features.

Response (200 OK):


```
{
  "success": true,
  "data": {
    "name": "Visual Array API",
    "currentVersion": "1.1.0",
    "description": "REST API for Visual Array system",
    "supportedVersions": ["1.0.0", "1.1.0"],
    "deprecatedVersions": [],
    "sunsetDate": "2025-12-31T23:59:59.000Z",
    "status": {
      "1.0.0": {
        "status": "deprecated",
        "canUse": true,
        "recommended": false,
        "sunsetDate": "2025-12-31T23:59:59.000Z",
        "breakingChanges": [],
        "newFeatures": []
      },
      "1.1.0": {
        "status": "current",
        "canUse": true,
        "recommended": true,
        "sunsetDate": null,
        "breakingChanges": [],
```

```
        "newFeatures": [
          "URL decoding for special characters",
          "Pagination support for list endpoints",
          "RS232 presets",
          "AV matrix routes/presets",
          "Video walls",
          "Advanced resource filtering",
          "Keywords search",
          "Tag-based filtering",
          "Simplified device type filtering (ENCODER/DECODER)",
          "Specialized AV route creation",
          "Specialized KVM route creation",
          "Stream type enum optimization (removed NONE)",
          "HDCP input and output configuration",
        ]
      }
    },
    "endpoints": {
      "1.0.0": {
        "api-info": "/api",
        "auth-login": "/api/v1/auth/login",
        "auth-logout": "/api/v1/auth/logout",
        "auth-validate": "/api/v1/auth/validate",
        "resources": "/api/v1/resource/resources",
        "resource": "/api/v1/resource/resource",
        "tx-stream-infos": "/api/v1/resource/txStreamInfos",
        "tx-stream-info": "/api/v1/resource/txStreamInfo",
        "rx-input-selection": "/api/v1/resource/rx/inputSelection",
        "tx-input-selection": "/api/v1/resource/tx/inputSelection",
        "rs232-send": "/api/v1/resource/rs232/send",
        "av-matrix-route": "/api/v1/av-matrix/route",
        "av-matrix-route-delete": "/api/v1/av-matrix/route",
        "av-matrix-route-to-all": "/api/v1/av-matrix/routeToAll",
        "av-matrix-preset-save": "/api/v1/av-matrix/preset/save",
        "av-matrix-preset-apply": "/api/v1/av-matrix/preset/apply/:presetName"
      },
      "1.1.0": {
        "api-info": "/api",
        "auth-login": "/api/v1/auth/login",
        "auth-logout": "/api/v1/auth/logout",
        "auth-validate": "/api/v1/auth/validate",
        "resources": "/api/v1/resource/resources",
        "resource": "/api/v1/resource/resource",
        "tx-stream-infos": "/api/v1/resource/txStreamInfos",
        "tx-stream-info": "/api/v1/resource/txStreamInfo",
        "rx-input-selection": "/api/v1/resource/rx/inputSelection",
        "tx-input-selection": "/api/v1/resource/tx/inputSelection",
        "rs232-presets": "/api/v1/resource/rs232/presets",
        "rs232-send": "/api/v1/resource/rs232/send",
        "hdcp-input": "/api/v1/resource/hdcp/input",
        "hdcp-output": "/api/v1/resource/hdcp/output",
        "tags": "/api/v1/resource/tags",
        "groups": "/api/v1/resource/groups",
        "av-matrix-routes": "/api/v1/av-matrix/routes",
```

```
        "av-matrix-presets": "/api/v1/av-matrix/presets",
        "av-matrix-route": "/api/v1/av-matrix/route",
        "av-matrix-av-route": "/api/v1/av-matrix/avRoute",
        "av-matrix-kvm-route": "/api/v1/av-matrix/kvmRoute",
        "av-matrix-data-route": "/api/v1/av-matrix/dataRoute",
        "av-matrix-route-delete": "/api/v1/av-matrix/route",
        "av-matrix-av-route-delete": "/api/v1/av-matrix/avRoute",
        "av-matrix-kvm-route-delete": "/api/v1/av-matrix/kvmRoute",
        "av-matrix-data-route-delete": "/api/v1/av-matrix/dataRoute",
        "av-matrix-route-to-all": "/api/v1/av-matrix/routeToAll",
        "av-matrix-route-to-all-av": "/api/v1/av-matrix/routeToAllAv",
        "av-matrix-route-to-all-data": "/api/v1/av-matrix/routeToAllData",
        "av-matrix-preset-save": "/api/v1/av-matrix/preset/save",
        "av-matrix-preset-apply": "/api/v1/av-matrix/preset/apply/:presetName",
        "video-walls": "/api/v1/video-wall/video-walls",
        "video-wall-set-state": "/api/v1/video-wall/setState"
      }
    },
    "features": {
      "1.0.0": [
        "User authentication",
        "Session management",
        "Device discovery",
        "Resource management",
        "Stream information",
        "Input selection",
        "RS232 control",
        "AV Matrix routing",
        "Preset management"
      ],
      "1.1.0": [
        "URL decoding for special characters",
        "Pagination support for list endpoints",
        "RS232 presets",
        "AV matrix routes/presets",
        "Video walls",
        "Advanced resource filtering",
        "Keywords search",
        "Tag-based filtering",
        "Simplified device type filtering (ENCODER/DECODER)",
        "Specialized AV route creation",
        "Specialized KVM route creation",
        "Stream type enum optimization (removed NONE)",
        "HDCP input and output configuration"
      ]
    },
    "documentation": "/api/docs",
    "migrationGuide": "/api/migration/1.0-to-1.1",
    "lastUpdated": "2025-08-08T00:00:00.000Z"
  },
  "message": "Success",
  "timestamp": "2025-01-27T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data`	Object	API information object
`data.name`	String	API name
`data.currentVersion`	String	Current API version
`data.description`	String	API description
`data.supportedVersions`	Array	List of supported API versions
`data.deprecatedVersions`	Array	List of deprecated versions (empty)
`data.sunsetDate`	Null	Sunset date for deprecated versions (null)
`data.status`	Object	Version status information
`data.endpoints`	Object	Available endpoints by version
`data.features`	Object	Available features by version
`data.documentation`	String	URL to API documentation
`data.migrationGuide`	Null	Migration guide URL (null)
`data.lastUpdated`	String	Last update timestamp

Resource Management

Manage devices (encoders and decoders) in the Visual Array system.

Get All Resources

Purpose: List all resources with optional filtering.

Endpoint: `GET /api/v1/resource/resources`

Description: Retrieves a list of all devices, including their status and configuration with pagination support.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Number of items per page (default: 25, max: 100)
`groupName`	String	No	Filter by group name
`keywords`	String	No	Filter by keywords (fuzzy match on label)
`tagName`	String	No	Filter by tag name
`deviceType`	String	No	Filter by device type (ENCODER or DECODER)

Request Examples:

```
GET /api/v1/resource/resources?page=1&limit=25
GET /api/v1/resource/resources?groupName=production&keywords=encoder
GET /api/v1/resource/resources?deviceType=ENCODER
GET /api/v1/resource/resources?deviceType=DECODER
GET /api/v1/resource/resources?tagName=production
GET /api/v1/resource/resources?keywords=encoder&deviceType=ENCODER&page=1&limit=10
```

Response (200 OK):


```
{
  "success": true,
  "data": [
    {
      "id": "resource-uuid",
      "label": "encoder-01",
      "model": "P-AVN-4",
      "tag": ["sports", "fav"],
      "group": "Room 1",
      "deviceInfo": {
        "type": "ENCODER",
        "state": "ONLINE/Started",
        "eth1_ip": "192.168.1.100",
        "eth2_ip": "192.168.1.101",
        "sfp_ip": "192.168.1.102",
        "fw_version": "1.2.3"
      },
      "tx": {
        "HC_Stream": {
          "videoStreamState": "ACTIVE",
          "audioStreamState": "ACTIVE"
        },
        "LC_Stream": {
          "videoStreamState": "ACTIVE",
          "audioStreamState": "ACTIVE"
        },
        "videoInput": {
          "source": "HDMI",
          "state": "ACTIVE",
          "hdcp_version": "HDCP 2.x"
        },
        "audioInput": {
          "source": "HDMI",
          "state": "ACTIVE"
        },
        "videoOutput": {
          "HDMI": {
            "enabled": "TRUE",
            "state": "ACTIVE",
```

```
            "hdcp_version": "HDCP 2.x"
          }
        },
        "audioOutput": {
          "HDMI": {
            "state": "ACTIVE"
          },
          "Analog": {
            "enabled": "TRUE",
            "state": "ACTIVE",
            "source": "HDMI"
          },
          "Dante": {
            "enabled": "TRUE",
            "state": "ACTIVE",
            "source": "HDMI"
          }
        }
      }
    },
    {
      "id": "resource-uuid-2",
      "label": "decoder-01",
      "model": "P-AVN-4",
      "tag": ["sports", "fav"],
      "group": "Room 1",
      "deviceInfo": {
        "type": "DECODER",
        "state": "ONLINE/Started",
        "eth1_ip": "192.168.1.100",
        "eth2_ip": "192.168.1.101",
        "fw_version": "1.3.0"
      },
      "rx": {
        "videoInput": {
          "source": "IP",
          "state": "ACTIVE",
          "hdcp_version": "HDCP 2.x"
        },
        "audioInput": {
          "source": "HDMI",
          "state": "ACTIVE"
        },
        "videoOutput": {
          "HDMI": {
            "enabled": "TRUE",
            "state": "ACTIVE",
            "hdcp_version": "HDCP 2.x"
          }
        },
        "audioOutput": {
          "HDMI": {
            "state": "ACTIVE"
          },
```

```
            "Analog": {
              "source": "HDMI",
              "state": "ACTIVE"
            }
          }
        }
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 25,
      "total": 150,
      "pages": 6
    },
    "message": "Success",
    "timestamp": "2024-01-01T12:00:00.000Z"
  }
```


Response Fields:

Field	Type	Description
`data`	Array	List of resource objects
`data[].id`	String	Unique identifier for the resource
`data[].label`	String	Human-readable name for the resource
`data[].model`	String	Model name of the device
`data[].tag`	Array	List of tags associated with the resource
`data[].group`	String	Group name the resource belongs to
`data[].deviceInfo`	Object	Device-specific information
`data[].deviceInfo.type`	String	Device type (ENCODER/DECODER)
`data[].deviceInfo.state`	String	Device state (ONLINE/Started, OFFLINE, etc.)
`data[].deviceInfo.eth1_ip`	String	IP address for Ethernet interface 1
`data[].deviceInfo.eth2_ip`	String	IP address for Ethernet interface 2
`data[].deviceInfo.sfp_ip`	String	IP address for SFP interface
`data[].deviceInfo.fw_version`	String	Firmware version of the device
`data[].tx`	Object	Transmitter-specific information (for encoders)
`data[].tx.HC_Stream`	Object	High Compressed stream information
`data[].tx.LC_Stream`	Object	Low Compressed stream information
`data[].tx.videoInput`	Object	Video input source information
`data[].tx.audioInput`	Object	Audio input source information

Field	Type	Description
`data[].tx.videoOutput`	Object	Video output information
`data[].tx.videoOutput.HDMI`	Object	HDMI video output information
`data[].tx.audioOutput`	Object	Audio output information
`data[].tx.audioOutput.HDMI`	Object	HDMI audio output information
`data[].tx.audioOutput.Analog`	Object	Analog audio output information
`data[].tx.audioOutput.Analog.source`	String	Audio source for analog output (HDMI)
`data[].tx.audioOutput.Dante`	Object	Dante audio output information
`data[].tx.audioOutput.Dante.source`	String	Audio source for Dante output (HDMI)
`data[].rx`	Object	Receiver-specific information (for decoders)
`data[].rx.videoInput`	Object	Video input source information
`data[].rx.audioInput`	Object	Audio input source information
`data[].rx.videoOutput`	Object	Video output information
`data[].rx.videoOutput.HDMI`	Object	HDMI video output information
`data[].rx.audioOutput`	Object	Audio output information
`data[].rx.audioOutput.HDMI`	Object	HDMI audio output information
`data[].rx.audioOutput.Analog`	Object	Analog audio output information
`data[].rx.audioOutput.Analog.source`	String	Audio source for analog output (HDMI)
`pagination`	Object	Pagination information
`pagination.page`	Number	Current page number
`pagination.limit`	Number	Items per page
`pagination.total`	Number	Total number of resources
`pagination.pages`	Number	Total number of pages

Enums:

- **Device Type**: `"ENCODER"` (includes MINI_ENCODER), `"DECODER"` (includes MINI_DECODER)
- **Device State**: `"ONLINE/Started"`, `"ONLINE/Stopped"`, `"OFFLINE"`
- **Stream State**: `"ACTIVE"`, `"INACTIVE"`
- **Video Input Source**: `"HDMI"`, `"USB"`, `"IP"`
- **Audio Input Source**: `"HDMI"`, `"USB"`, `"DANTE"`, `"ANALOG"`, `"IP"`
- **Output State**: `"TRUE"`, `"FALSE"` (for enabled field)

Note: The response structure has been updated to include detailed device information, stream states, and input/output configurations. Encoders include `tx` (transmit) information, while decoders include `rx` (receive) information.

Get Single Resource

Purpose: Retrieve details for a specific resource.

Endpoint: `GET /api/v1/resource/resource`

Description: Returns information about a single device.

Query Parameters:

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the resource

Request Example:

```
GET /api/v1/resource/resource?resourceName=encoder-01
```

Response (200 OK):


```
{
  "success": true,
  "data": {
    "id": "resource-uuid",
    "label": "encoder-01",
    "model": "P-AVN-4/2/-2E/-2D",
    "tag": ["sports", "fav"],
    "group": "Room 1",
    "deviceInfo": {
      "type": "ENCODER",
      "state": "ONLINE/Started",
      "eth1_ip": "192.168.1.100",
      "eth2_ip": "192.168.1.101",
      "sfp_ip": "192.168.1.102",
      "fw_version": "1.2.3"
    },
    "tx": {
      "HC_Stream": {
        "videoStreamState": "ACTIVE",
        "audioStreamState": "ACTIVE"
      },
      "LC_Stream": {
        "videoStreamState": "ACTIVE",
        "audioStreamState": "ACTIVE"
      },
      "videoInput": {
        "source": "HDMI",
        "state": "ACTIVE",
        "hdcp_version": "HDCP 2.x"
      },
      "audioInput": {
```

```
        "source": "HDMI",
        "state": "ACTIVE"
      },
      "videoOutput": {
        "HDMI": {
          "enabled": "TRUE",
          "state": "ACTIVE",
          "hdcp_version": "HDCP 2.x"
        }
      },
      "audioOutput": {
        "HDMI": {
          "state": "ACTIVE"
        },
        "Analog": {
          "enabled": "TRUE",
          "state": "ACTIVE",
          "source": "HDMI"
        },
        "Dante": {
          "enabled": "TRUE",
          "state": "ACTIVE",
          "source": "HDMI"
        }
      }
    }
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields: Same structure as Get All Resources response (single resource object).

Decoder Response Example:


```
{
  "success": true,
  "data": {
    "id": "decoder-uuid",
    "label": "decoder-01",
    "model": "P-AVN-4/2/-2E/-2D",
    "tag": ["display", "main"],
    "group": "Room 2",
    "deviceInfo": {
      "type": "DECODER",
      "state": "ONLINE/Started",
      "eth1_ip": "192.168.1.200",
      "eth2_ip": "192.168.1.201",
      "sfp_ip": "192.168.1.202",
      "fw_version": "1.2.3"
    },
    "rx": {
```

```
        "videoInput": {
          "source": "IP",
          "state": "ACTIVE",
          "hdcp_version": "HDCP 2.x"
        },
        "audioInput": {
          "source": "HDMI",
          "state": "ACTIVE"
        },
        "videoOutput": {
          "HDMI": {
            "enabled": "TRUE",
            "state": "ACTIVE",
            "hdcp_version": "HDCP 2.x"
          }
        },
        "audioOutput": {
          "HDMI": {
            "state": "ACTIVE"
          },
          "Analog": {
            "source": "HDMI",
            "state": "ACTIVE"
          }
        }
      }
    },
    "message": "Success",
    "timestamp": "2024-01-01T12:00:00.000Z"
  }
```


Get TX Stream Infos

Purpose: Retrieve stream information for all encoders.

Endpoint: `GET /api/v1/resource/txStreamInfos`

Description: Returns stream details for all transmitter devices with pagination support.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Number of items per page (default: 25, max: 100)

Request Examples:

```
GET /api/v1/resource/txStreamInfos
GET /api/v1/resource/txStreamInfos?page=1&limit=50
```

Response (200 OK):

```
{
  "success": true,
  "data": [
    {
      "label": "Encoder-1",
      "videoInputFormat": "3840x2160p 60fps",
      "audioInputFormat": "PCM/RAW",
      "HC": {
        "videoIp": "192.168.1.100",
        "audioIp": "192.168.1.101",
        "dataIp": "192.168.1.102",
        "videoOutputFormat": "1920x1080p 60fps",
        "codec": "H264",
        "videoStreamState": "ACTIVE",
        "audioStreamState": "ACTIVE"
      },
      "LC": {
        "videoIp": "192.168.1.103",
        "audioIp": "192.168.1.104",
        "dataIp": "192.168.1.105",
        "videoOutputFormat": "1280x720p 30fps",
        "codec": "H264",
        "videoStreamState": "ACTIVE",
        "audioStreamState": "ACTIVE"
      }
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
    "total": 50,
    "pages": 2
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data`	Array	List of encoder stream info objects
`data[].label`	String	Encoder device name
`data[].videoInputFormat`	String	Video input format (e.g., "3840x2160p 60fps")
`data[].audioInputFormat`	String	Audio input format (e.g., "PCM/RAW")
`data[].HC`	Object	High Compressed stream information

Field	Type	Description
`data[].HC.videoIp`	String	Video stream IP address
`data[].HC.audioIp`	String	Audio stream IP address
`data[].HC.dataIp`	String	Data stream IP address
`data[].HC.videoOutputFormat`	String	Video output format (e.g., "1920x1080p 60fps")
`data[].HC.codec`	String	Video codec (H264, H265, JPEG)
`data[].HC.videoStreamState`	String	Video stream state (ACTIVE, INACTIVE, ERROR)
`data[].HC.audioStreamState`	String	Audio stream state (ACTIVE, INACTIVE, ERROR)
`data[].LC`	Object	Low Compressed stream information
`data[].LC.videoIp`	String	Video stream IP address
`data[].LC.audioIp`	String	Audio stream IP address
`data[].LC.dataIp`	String	Data stream IP address
`data[].LC.videoOutputFormat`	String	Video output format (e.g., "1280x720p 30fps")
`data[].LC.codec`	String	Video codec (H264, H265, JPEG)
`data[].LC.videoStreamState`	String	Video stream state (ACTIVE, INACTIVE, ERROR)
`data[].LC.audioStreamState`	String	Audio stream state (ACTIVE, INACTIVE, ERROR)
`pagination`	Object	Pagination information
`pagination.page`	Number	Current page number
`pagination.limit`	Number	Items per page
`pagination.total`	Number	Total number of encoders
`pagination.pages`	Number	Total number of pages

Enums:

- **Codec**: `"H264"`, `"H265"`, `"JPEG"`
- **Stream State**: `"ACTIVE"`, `"INACTIVE"`, `"ERROR"`

Get TX Stream Info

Purpose: Retrieve stream information for a specific encoder.

Endpoint: `GET /api/v1/resource/txStreamInfo`

Description: Returns stream details for a single transmitter device.

Query Parameters:

Parameter	Type	Required	Description

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the encoder

Request Example:

```
GET /api/v1/resource/txStreamInfo?resourceName=encoder-01
```

Response (200 OK):


```
{
  "success": true,
  "data": {
    "label": "Encoder-1",
    "videoInputFormat": "3840x2160p 60fps",
    "audioInputFormat": "PCM/RAW",
    "HC": {
      "videoIp": "192.168.1.100",
      "audioIp": "192.168.1.101",
      "dataIp": "192.168.1.102",
      "videoOutputFormat": "1920x1080p 60fps",
      "codec": "H264",
      "videoStreamState": "ACTIVE",
      "audioStreamState": "ACTIVE"
    },
    "LC": {
      "videoIp": "192.168.1.103",
      "audioIp": "192.168.1.104",
      "dataIp": "192.168.1.105",
      "videoOutputFormat": "1280x720p 30fps",
      "codec": "H264",
      "videoStreamState": "ACTIVE",
      "audioStreamState": "ACTIVE"
    }
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields: Same structure as Get TX Stream Infos response (single encoder object).

Select RX Input

Purpose: Change the input source for a receiver.

Endpoint: `POST /api/v1/resource/rx/inputSelection`

Description: Sets the video or audio input source for a receiver device.

Query Parameters:

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the receiver
`type`	String	Yes	Input type (`video`, `audio`)

Request Body:

```
{
  "inputSelection": "IP"
}
```

Valid Input Selections:

- **Video**: `"IP"`, `"HDMI"`, `"USB"`
- **Audio**: `"IP"`, `"HDMI"`, `"DANTE"`, `"ANALOG"`, `"USB"`

Request Example:

```
POST /api/v1/resource/rx/inputSelection?resourceName=decoder-01&type=video
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "updatedResources": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.updatedResources`	Number	Number of resources that were updated

Select TX Input

Purpose: Change the input source for a transmitter.

Endpoint: `POST /api/v1/resource/tx/inputSelection`

Description: Sets the video or audio input source for a transmitter device.

Query Parameters:

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the transmitter
`type`	String	Yes	Input type (`video`, `audio`)

Request Body:

```
{
  "inputSelection": "HDMI"
}
```

Valid Input Selections:

- **Video**: `"HDMI"`, `"USB"`
- **Audio**: `"HDMI"`, `"DANTE"`, `"ANALOG"`, `"USB"`

Request Example:

```
POST /api/v1/resource/tx/inputSelection?resourceName=encoder-01&type=video
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "updatedResources": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.updatedResources`	Number	Number of resources that were updated

Get RS232 Presets

Purpose: Retrieve available RS232 preset commands for a device.

Endpoint: `GET /api/v1/resource/rs232/presets`

Description: Returns a list of predefined RS232 commands available for the specified device.

Query Parameters:

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the target resource

Request Example:

```
GET /api/v1/resource/rs232/presets?resourceName=decoder-01
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "resourceName": "decoder-01",
    "presets": [
      {
        "name": "power_on",
        "rawCommand": "PWR ON",
        "hexMode": false
      },
      {
        "name": "power_off",
        "rawCommand": "PWR OFF",
        "hexMode": false
      }
    ]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.resourceName`	String	Name of the target resource
`data.presets`	Array	List of available RS232 preset commands
`data.presets[].name`	String	Name of the preset command
`data.presets[].rawCommand`	String	Raw RS232 command string
`data.presets[].hexMode`	Boolean	Whether the command uses hex mode

Send RS232 Preset Command

Purpose: Execute a preset RS232 command on a device.

Endpoint: `POST /api/v1/resource/rs232/send`

Description: Sends a predefined RS232 command to a specified device.

Query Parameters:

Parameter	Type	Required	Description
`presetName`	String	Yes	Name of the preset command
`resourceName`	String	Yes	Name of the target resource

Request Example:

```
POST /api/v1/resource/rs232/send?presetName=power_on&resourceName=decoder-01
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "presetName": "power_on",
    "resourceName": "decoder-01",
    "commandInstance": "command-data",
    "updatedResources": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.presetName`	String	Name of the preset command that was executed
`data.resourceName`	String	Name of the target resource
`data.commandInstance`	String	Command instance identifier
`data.updatedResources`	Number	Number of resources that were updated

Configure HDCP Input Settings

Purpose: Configure HDCP input settings for a specific resource.

Endpoint: `POST /api/v1/resource/hdcp/input`

Description: Updates HDCP input configuration for HDMI inputs on encoders and decoders.

Query Parameters:

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the target resource

Request Body:

```
{
  "hdcpCapability": "AUTO"
}
```

Request Fields:

Field	Type	Required	Description
`hdcpCapability`	String	Yes	HDCP capability type (AUTO, OFF)

Response (200 OK):

```
{
  "success": true,
  "data": {
    "resourceName": "encoder-01",
    "updatedFields": ["hdcpCapability"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.resourceName`	String	Name of the target resource
`data.updatedFields`	Array	List of fields that were updated

Configure HDCP Output Settings

Purpose: Configure HDCP output settings for a specific resource.

Endpoint: `POST /api/v1/resource/hdcp/output`

Description: Updates HDCP output configuration for HDMI outputs on encoders and decoders. The backend automatically determines whether to apply RX (receiver) or TX (transmitter) settings based on the resource type; no `type` parameter is needed.

Query Parameters:

Parameter	Type	Required	Description

Parameter	Type	Required	Description
`resourceName`	String	Yes	Name of the target resource

Request Body:

```
{
  "hdcpType": "HDCP_FOLLOW_INPUT"
}
```

Request Fields:

Field	Type	Required	Description
`hdcpType`	String	Yes	HDCP output type (HDCP_FOLLOW_INPUT, HDCP_FORCE_HIGHEST)

Response (200 OK):

```
{
  "success": true,
  "data": {
    "resourceName": "encoder-01",
    "updatedFields": ["hdmiOutputTransmitter"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields: Same structure as Configure HDCP Input Settings response.

Get Tags

Purpose: Retrieve all available tags for resource filtering and management.

Endpoint: `GET /api/v1/resource/tags`

Description: Returns a paginated list of all tags that can be used for resource filtering and organization.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Number of items per page (default: 25, max: 100)

Request Examples:

```
GET /api/v1/resource/tags
GET /api/v1/resource/tags?page=1&limit=50
```

Response (200 OK):


```
{
  "success": true,
  "data": [
    {
      "id": "tag-uuid-1",
      "label": "production"
    },
    {
      "id": "tag-uuid-2",
      "label": "testing"
    },
    {
      "id": "tag-uuid-3",
      "label": "backup"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
    "total": 3,
    "pages": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields:

Field	Type	Description
`data`	Array	List of tag objects
`data[].id`	String	Unique tag identifier
`data[].label`	String	Tag name/label
`pagination`	Object	Pagination information
`pagination.page`	Number	Current page number
`pagination.limit`	Number	Items per page
`pagination.total`	Number	Total number of tags
`pagination.pages`	Number	Total number of pages

Get Resource Groups

Purpose: Retrieve all available resource groups for organization and filtering.

Endpoint: `GET /api/v1/resource/groups`

Description: Returns a paginated list of all resource groups that can be used for organizing and filtering resources.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Number of items per page (default: 25, max: 100)

Request Examples:

```
GET /api/v1/resource/groups
GET /api/v1/resource/groups?page=1&limit=50
```

Response (200 OK):


```
{
  "success": true,
  "data": [
    {
      "id": "group-uuid-1",
      "label": "Production Room A"
    },
    {
      "id": "group-uuid-2",
      "label": "Testing Lab"
    },
    {
      "id": "group-uuid-3",
      "label": "Conference Hall"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
    "total": 3,
    "pages": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data`	Array	List of resource group objects
`data[].id`	String	Unique resource group identifier
`data[].label`	String	Resource group name/label
`pagination`	Object	Pagination information
`pagination.page`	Number	Current page number
`pagination.limit`	Number	Items per page
`pagination.total`	Number	Total number of resource groups
`pagination.pages`	Number	Total number of pages

AV Matrix Management

Manage routing between transmitters and receivers.

Stream Type Enum

Value	Description
`LC`	Low Compressed stream
`HC`	High Compressed stream

Get Routes

Purpose: Retrieve all AV matrix routes.

Endpoint: `GET /api/v1/av-matrix/routes`

Description: Returns a paginated list of all current AV matrix routes.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Number of items per page (default: 25, max: 100)

Request Example:

```
GET /api/v1/av-matrix/routes?page=1&limit=25
```

Response (200 OK):

```
{
  "success": true,
  "data": [
    {
      "id": "route-uuid-1",
      "senderId": "sender-uuid",
      "senderName": "transmitter1",
      "receiverId": "receiver-uuid",
      "receiverName": "receiver1",
      "linkType": "VIDEO_HC"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
    "total": 25,
    "pages": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields:

Field	Type	Description
`data`	Array	List of routes
`data[].id`	String	Unique identifier for the route
`data[].senderId`	String	Unique identifier for the sender device
`data[].senderName`	String	Name of the sender device
`data[].receiverId`	String	Unique identifier for the receiver device
`data[].receiverName`	String	Name of the receiver device
`data[].linkType`	String	Type of link (VIDEO_HC, AUDIO_LC, USB, etc.)

Get Presets

Purpose: Retrieve all saved AV matrix presets.

Endpoint: `GET /api/v1/av-matrix/presets`

Description: Returns a paginated list of all saved AV matrix presets.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)

Parameter	Type	Required	Description
`limit`	Number	No	Number of items per page (default: 25, max: 100)

Request Example:

```
GET /api/v1/av-matrix/presets?page=1&limit=25
```

Response (200 OK):

```
{
  "success": true,
  "data": [
    {
      "presetId": "preset-uuid-1",
      "presetName": "my_preset_001"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
    "total": 10,
    "pages": 1
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data`	Array	List of presets
`data[].presetId`	String	Unique identifier for the preset
`data[].presetName`	String	Name of the preset

Create Route

Purpose: Establish a route between a transmitter and receiver.

Endpoint: `POST /api/v1/av-matrix/route`

Description: Creates a connection with specified video, audio, USB, and RS232 settings.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1",
  "videoType": "HC",
  "audioType": "LC",
  "usb": true,
  "rs232": false
}
```

Parameters:

Parameter	Type	Required	Description
rxName	String	Yes	Receiver device name
txName	String	Yes	Transmitter device name
videoType	String	No	Video stream type (HC or LC only)
audioType	String	No	Audio stream type (HC or LC only)
usb	Boolean	No	Include USB connection
rs232	Boolean	No	Include RS232 connection

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "createdLinkTypes": ["VIDEO_HC", "AUDIO_LC", "USB"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
data.txName	String	Name of the transmitter device
data.rxName	String	Name of the receiver device
data.createdLinkTypes	Array	List of link types that were created

Create Data Route

Purpose: Create a data route with RS232 only.

Endpoint: `POST /api/v1/av-matrix/dataRoute`

Description: Creates a specialized data route that supports RS232 connections only. This endpoint is optimized for data-specific routing scenarios.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	String	Yes	Receiver device name
`txName`	String	Yes	Transmitter device name

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "createdLinkTypes": ["RS232"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.createdLinkTypes`	Array	List of link types that were created (RS232)

Error Responses:

- **400 Bad Request**: Invalid parameters provided

```
{
  "success": false,
```

```
    "message": "No valid link types specified",
    "data": null,
    "timestamp": "2024-01-01T12:00:00.000Z"
  }
```

Create AV Route

Purpose: Create an AV route with video and audio only.

Endpoint: `POST /api/v1/av-matrix/avRoute`

Description: Creates a specialized AV route that supports video and audio connections only. This endpoint is optimized for AV-specific routing scenarios.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1",
  "streamType": "HC"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	String	Yes	Receiver device name
`txName`	String	Yes	Transmitter device name
`streamType`	String	No	Stream type (HC or LC only) - both video and audio will use this type

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "createdLinkTypes": ["VIDEO_HC", "AUDIO_LC"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.createdLinkTypes`	Array	List of link types that were created (Video and Audio)

Error Responses:

- **400 Bad Request**: Invalid parameters provided

```
{
  "success": false,
  "message": "No valid link types specified",
  "data": null,
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Create KVM Route

Purpose: Create a KVM route with video, audio, and USB support.

Endpoint: `POST /api/v1/av-matrix/kvmRoute`

Description: Creates a specialized KVM route that supports video, audio, and USB connections. USB connection is always included by default for KVM functionality. This endpoint is optimized for KVM-specific routing scenarios.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1",
  "streamType": "HC"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	String	Yes	Receiver device name
`txName`	String	Yes	Transmitter device name
`streamType`	String	No	Stream type (HC or LC only) - both video and audio will use this type

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "createdLinkTypes": ["VIDEO_HC", "AUDIO_LC", "USB"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.createdLinkTypes`	Array	List of link types that were created (Video, Audio, and USB)

Error Responses:

- **400 Bad Request**: Invalid parameters provided

```
{
  "success": false,
  "message": "No valid link types specified",
  "data": null,
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Delete Route

Purpose: Remove a route between a transmitter and receiver.

Endpoint: `DELETE /api/v1/av-matrix/route`

Description: Deletes specified connections between devices.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1",
  "videoType": "HC",
  "audioType": "LC",
  "usb": true,
```

```
  "rs232": false
}
```

Parameters: Same as Create Route.

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "deletedLinkTypes": ["VIDEO_HC", "AUDIO_LC", "USB"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.deletedLinkTypes`	Array	List of link types that were deleted

Delete Data Route

Purpose: Remove a data route (RS232 only).

Endpoint: `DELETE /api/v1/av-matrix/dataRoute`

Description: Deletes RS232 connections between devices. This endpoint is optimized for data-specific routing scenarios.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	String	Yes	Receiver device name

Parameter	Type	Required	Description
`txName`	String	Yes	Transmitter device name

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "deletedLinkTypes": ["RS232"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.deletedLinkTypes`	Array	List of link types that were deleted (RS232)

Error Responses:

- **400 Bad Request**: Invalid parameters provided

```
{
  "success": false,
  "message": "No valid link types specified",
  "data": null,
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Delete AV Route

Purpose: Remove an AV route (video and audio only).

Endpoint: `DELETE /api/v1/av-matrix/avRoute`

Description: Deletes video and audio connections between devices. This endpoint is optimized for AV-specific routing scenarios.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1",
  "streamType": "HC"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	String	Yes	Receiver device name
`txName`	String	Yes	Transmitter device name
`streamType`	String	No	Stream type (HC or LC only) - both video and audio will use this type

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "deletedLinkTypes": ["VIDEO_HC", "AUDIO_HC"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.deletedLinkTypes`	Array	List of link types that were deleted (Video and Audio)

Error Responses:

- **400 Bad Request**: Invalid parameters provided

```
{
  "success": false,
  "message": "No valid link types specified",
  "data": null,
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Delete KVM Route

Purpose: Remove a KVM route (video, audio, and USB).

Endpoint: `DELETE /api/v1/av-matrix/kvmRoute`

Description: Deletes video, audio, and USB connections between devices. USB connection is always included by default for KVM functionality.

Request Body:

```
{
  "rxName": "receiver1",
  "txName": "transmitter1",
  "streamType": "HC"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	String	Yes	Receiver device name
`txName`	String	Yes	Transmitter device name
`streamType`	String	No	Stream type (HC or LC only) - both video and audio will use this type

Response (200 OK):

```
{
  "success": true,
  "data": {
    "txName": "transmitter1",
    "rxName": "receiver1",
    "deletedLinkTypes": ["VIDEO_HC", "AUDIO_HC", "USB"]
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.txName`	String	Name of the transmitter device
`data.rxName`	String	Name of the receiver device
`data.deletedLinkTypes`	Array	List of link types that were deleted (Video, Audio, and USB)

Error Responses:

- **400 Bad Request**: Invalid parameters provided

```
{
  "success": false,
  "message": "No valid link types specified",
  "data": null,
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Route to All

Purpose: Route a transmitter to multiple receivers.

Endpoint: `POST /api/v1/av-matrix/routeToAll`

Description: Creates connections from one transmitter to multiple receivers.

Request Body:

```
{
  "rxName": ["receiver1", "receiver2", "receiver3"],
  "txName": "transmitter1",
  "videoType": "HC",
  "audioType": "LC",
  "rs232": true
}
```

Minimal Request:

```
{
  "rxName": ["receiver1", "receiver2", "receiver3"],
  "txName": "transmitter1",
  "videoType": "HC"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	Array	Yes	List of receiver device names
`txName`	String	Yes	Transmitter device name
`videoType`	String	Yes	Video stream type (LC, HC)
`audioType`	String	No	Audio stream type (LC, HC)

Parameter	Type	Required	Description
rs232	Boolean	No	Include RS232 connection

Response (200 OK):


```
{
  "success": true,
  "data": {
    "routes": [
      {
        "txName": "transmitter1",
        "rxName": "receiver1",
        "createdLinkTypes": ["VIDEO_HC", "RS232"]
      }
    ],
    "failedRoutes": [
      {
        "receiver": "receiver2",
        "error": "Device not found"
      }
    ],
    "summary": {
      "transmitter": "transmitter1",
      "receiverCount": 2,
      "failedCount": 1,
      "videoType": "HC",
      "audioType": "LC",
      "rs232": true,
      "totalRoutes": 3
    }
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields:

Field	Type	Description
data.routes	Array	List of successfully created routes
data.routes[].txName	String	Name of the transmitter device
data.routes[].rxName	String	Name of the receiver device
data.routes[].createdLinkTypes	Array	List of link types that were created
data.failedRoutes	Array	List of routes that failed to create (optional)
data.failedRoutes[].receiver	String	Name of the receiver that failed
data.failedRoutes[].error	String	Error message for the failed route

Field	Type	Description
`data.summary`	Object	Summary of the operation
`data.summary.transmitter`	String	Name of the transmitter device
`data.summary.receiverCount`	Number	Number of successfully created routes
`data.summary.failedCount`	Number	Number of routes that failed to create
`data.summary.videoType`	String	Video type used for the routes (if applicable)
`data.summary.audioType`	String	Audio type used for the routes (if applicable)
`data.summary.rs232`	Boolean	Whether RS232 was included (if applicable)
`data.summary.totalRoutes`	Number	Total number of routes attempted

Route to All Data

Purpose: Route data to all devices (RS232 only).

Endpoint: `POST /api/v1/av-matrix/routeToAllData`

Description: Creates data connections from one transmitter to multiple receivers. RS232 connection is always included by default for data functionality.

Request Body:

```
{
  "rxName": ["receiver1", "receiver2", "receiver3"],
  "txName": "transmitter1"
}
```

Parameters:

Parameter	Type	Required	Description
`rxName`	Array	Yes	List of receiver device names
`txName`	String	Yes	Transmitter device name

Response (200 OK):

```
{
  "success": true,
  "data": {
    "routes": [
      {
        "txName": "transmitter1",
        "rxName": "receiver1",
        "createdLinkTypes": ["RS232"]
```

```
        }
      ],
      "failedRoutes": [
        {
          "receiver": "receiver2",
          "error": "Device not found"
        }
      ],
      "summary": {
        "transmitter": "transmitter1",
        "receiverCount": 2,
        "failedCount": 1,
        "totalRoutes": 3
      }
    },
    "message": "Success",
    "timestamp": "2024-01-01T12:00:00.000Z"
  }
```

Response Fields:

Field	Type	Description
`data.routes`	Array	List of successfully created routes
`data.routes[].txName`	String	Name of the transmitter device
`data.routes[].rxName`	String	Name of the receiver device
`data.routes[].createdLinkTypes`	Array	List of link types that were created (RS232)
`data.failedRoutes`	Array	List of routes that failed to create (optional)
`data.failedRoutes[].receiver`	String	Name of the receiver that failed
`data.failedRoutes[].error`	String	Error message for the failed route
`data.summary`	Object	Summary of the operation
`data.summary.transmitter`	String	Name of the transmitter device
`data.summary.receiverCount`	Number	Number of successfully created routes
`data.summary.failedCount`	Number	Number of routes that failed to create
`data.summary.totalRoutes`	Number	Total number of routes attempted

Route to All AV

Purpose: Route AV to all devices (video and audio only).

Endpoint: `POST /api/v1/av-matrix/routeToAllAv`

Description: Creates AV connections from one transmitter to multiple receivers.

Request Body:

```
{
  "rxName": ["receiver1", "receiver2", "receiver3"],
  "txName": "transmitter1",
  "streamType": "HC"
}
```

Parameters:

Parameter	Type	Required	Description
rxName	Array	Yes	List of receiver device names
txName	String	Yes	Transmitter device name
streamType	String	No	Stream type (HC or LC only) - both video and audio will use this type

Response (200 OK):

```
{
  "success": true,
  "data": {
    "routes": [
      {
        "txName": "transmitter1",
        "rxName": "receiver1",
        "createdLinkTypes": ["VIDEO_HC", "AUDIO_HC"]
      }
    ],
    "failedRoutes": [
      {
        "receiver": "receiver2",
        "error": "Device not found"
      }
    ],
    "summary": {
      "transmitter": "transmitter1",
      "receiverCount": 2,
      "failedCount": 1,
      "streamType": "HC",
      "totalRoutes": 3
    }
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description

Field	Type	Description
`data.routes`	Array	List of successfully created routes
`data.routes[].txName`	String	Name of the transmitter device
`data.routes[].rxName`	String	Name of the receiver device
`data.routes[].createdLinkTypes`	Array	List of link types that were created (Video and Audio)
`data.failedRoutes`	Array	List of routes that failed to create (optional)
`data.failedRoutes[].receiver`	String	Name of the receiver that failed
`data.failedRoutes[].error`	String	Error message for the failed route
`data.summary`	Object	Summary of the operation
`data.summary.transmitter`	String	Name of the transmitter device
`data.summary.receiverCount`	Number	Number of successfully created routes
`data.summary.failedCount`	Number	Number of routes that failed to create
`data.summary.streamType`	String	Stream type used for the routes (if applicable)
`data.summary.totalRoutes`	Number	Total number of routes attempted

Save Preset

Purpose: Save the current AV matrix configuration.

Endpoint: `POST /api/v1/av-matrix/preset/save`

Description: Stores the current routing configuration as a named preset.

Request Body:

```
{
  "presetName": "my_preset_001",
  "userName": "admin"
}
```

Parameters:

Parameter	Type	Required	Description
`presetName`	String	Yes	Name for the preset
`userName`	String	Yes	User name for filtering

Response (200 OK):

```
{
  "success": true,
  "data": {
    "presetLabel": "my_preset_001",
    "routeCount": 12,
    "userName": "admin",
    "createdAt": "2024-01-01T12:00:00.000Z"
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields:

Field	Type	Description
`data.presetLabel`	String	Name of the saved preset
`data.routeCount`	Number	Number of routes included in the preset
`data.userName`	String	Name of the user who saved the preset
`data.createdAt`	String	Timestamp when the preset was created

Apply Preset

Purpose: Restore a saved AV matrix configuration.

Endpoint: `POST /api/v1/av-matrix/preset/apply/{presetName}`

Description: Applies a named preset to the AV matrix.

Parameters:

Parameter	Type	Required	Description
`presetName`	String	Yes	Name of the preset (path)

Response (200 OK):


```
{
  "success": true,
  "data": {
    "presetName": "my_preset_001"
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```


Response Fields:

Field	Type	Description
`data.presetName`	String	Name of the preset that was applied

Video Wall Management

Control video wall configurations.

Get Video Walls

Purpose: Retrieve all video walls.

Endpoint: `GET /api/v1/video-wall/video-walls`

Description: Returns a paginated list of all video walls in the system.

Query Parameters:

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Number of items per page (default: 25, max: 100)

Request Example:

```
GET /api/v1/video-wall/video-walls?page=1&limit=25
```

Response (200 OK):


```
{
  "success": true,
  "data": [
    {
      "videoWallId": "wall-uuid-1",
      "videoWallName": "wall1"
    },
    {
      "videoWallId": "wall-uuid-2",
      "videoWallName": "wall2"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
    "total": 2,
    "pages": 1
  },
  "message": "Success",
```

```
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data`	Array	List of video walls
`data[].videoWallId`	String	Unique identifier for the video wall
`data[].videoWallName`	String	Name of the video wall

Set Video Wall State

Purpose: Enable or disable a video wall.

Endpoint: `POST /api/v1/video-wall/setState`

Description: Sets the operational state of a video wall.

Query Parameters:

Parameter	Type	Required	Description
`videoWallName`	String	Yes	Name of the video wall
`state`	String	Yes	State (`ENABLED`, `DISABLED`)

Request Example:

```
POST /api/v1/video-wall/setState?videoWallName=wall1&state=ENABLED
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "videoWallName": "wall1",
    "state": "ENABLED",
    "videoWallId": "wall-uuid",
    "hasDuplicates": false
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Response Fields:

Field	Type	Description
`data.videoWallName`	String	Name of the video wall
`data.state`	String	Current state of the video wall (ENABLED/DISABLED)
`data.videoWallId`	String	Unique identifier for the video wall
`data.hasDuplicates`	Boolean	Indicates if there are duplicate configurations

Enums:

- **Video Wall State**: `"ENABLED"`, `"DISABLED"`

Common Response Fields

Most API endpoints return consistent response structures. The following fields are common across all endpoints:

Standard Success Response Fields

Field	Type	Description
`success`	Boolean	Always `true` for successful responses
`data`	Object	Response data specific to the endpoint
`timestamp`	String	ISO 8601 timestamp of the response

Example Success Response:

```
{
  "success": true,
  "data": {
    // Endpoint-specific data
  },
  "message": "Success",
  "timestamp": "2024-01-01T00:00:00.000Z"
}
```

Standard Error Response Fields

Field	Type	Description
`success`	Boolean	Always `false` for error responses
`data`	Object	Error details (may be null)
`message`	String	Human-readable error message
`timestamp`	String	ISO 8601 timestamp of the response

Example Error Response:

```
{
  "success": false,
  "data": null,
  "message": "Operation failed",
  "timestamp": "2024-01-01T00:00:00.000Z"
}
```

Pagination Fields (for list endpoints)

Field	Type	Description
`pagination.page`	Number	Current page number
`pagination.limit`	Number	Number of items per page
`pagination.total`	Number	Total number of items
`pagination.pages`	Number	Total number of pages

Error Handling

All endpoints return consistent error responses.

HTTP Status Codes:

Code	Description
200	Success
400	Bad Request
401	Unauthorized
404	Not Found
429	Too Many Requests
500	Internal Server Error

Error Response Format:

```
{
  "success": false,
  "data": null,
  "message": "Detailed error message",
  "timestamp": "2024-01-01T00:00:00.000Z"
}
```

Common Error Types:

- **Authentication Errors**: When login credentials are invalid or session has expired

- **Device Type Mismatch**: When trying to access RX endpoints on TX devices or vice versa
- **Validation Errors**: When request data fails validation
- **Device Not Found**: When device is not available
- **Rate Limit Exceeded**: When too many requests are made in a short time period
- **Internal Server Errors**: When unexpected server-side errors occur

Error Handling Best Practices:

1. **Check Success Field**: Always verify the `success` field in responses
2. **Handle HTTP Status Codes**: Implement proper handling for different HTTP status codes
3. **User-Friendly Messages**: Display user-friendly error messages based on the `message` field
4. **Retry Logic**: Implement appropriate retry logic for transient errors (5xx)
5. **Logging**: Log detailed error information for debugging purposes

Response Headers

Only the API discovery endpoint (`GET /api`) includes version headers. Other endpoints may omit these headers.

```
API-Version: 1.1.0
API-Current-Version: 1.1.0
API-Name: Visual Array API
API-Documentation: /api/docs
API-Status: current
API-Recommendation: Use v1.1.0 for development
```

Rate Limiting

The API enforces rate limits to prevent abuse. The default limit is 100 requests per minute. Exceeding this returns a `429 Too Many Requests` response.

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 99
X-RateLimit-Reset: 1631234567
```

Best Practices

Authentication & Security

1. **Authentication**: Always login first to obtain a valid session ID
2. **Session Management**: Use the session ID in subsequent requests and logout when done
3. **HTTPS Usage**: Always use HTTPS in production environments
4. **Secret Management**: Store API credentials securely and rotate them regularly

API Usage

5. **Device Type Awareness**: Always use the correct device type endpoints (RX/TX)

6. **Input Validation**: Validate input IDs and data before sending requests
7. **Error Handling**: Check the `success` field and handle errors gracefully
8. **Rate Limiting**: Respect rate limits and implement appropriate backoff strategies

Device Management

9. **Preset Usage**: Use presets for common operations
10. **Monitoring**: Regularly check device status and network connectivity
11. **Resource Cleanup**: Properly close sessions and clean up resources
12. **Logging**: Implement comprehensive logging for debugging and monitoring

Performance & Reliability

13. **Pagination**: Use pagination for large datasets to improve performance
14. **Caching**: Implement appropriate caching strategies for frequently accessed data
15. **Connection Management**: Reuse HTTP connections when possible
16. **Timeout Handling**: Set appropriate timeouts for different types of requests

Response Handling

17. **Success Field Check**: Always verify the `success` field in responses
18. **Data Field Access**: Access response data through the `data` field
19. **Timestamp Usage**: Use timestamps for logging and debugging
20. **Message Display**: Display user-friendly messages from the `message` field

Pagination

The API supports pagination for list endpoints to efficiently handle large datasets. Pagination is available in version 1.1.0 and later.

Pagination Parameters

Parameter	Type	Required	Description
`page`	Number	No	Page number for pagination (default: 1)
`limit`	Number	No	Items per page (default: 25, max: 100)

Pagination Response Format

All paginated endpoints return a consistent response format:

```
{
  "success": true,
  "data": [
    // Array of items for the current page
  ],
  "pagination": {
    "page": 1,
    "limit": 25,
```

```
    "total": 150,
    "pages": 3
  }
}
```

Pagination Response Fields

Field	Type	Description
`page`	Number	Current page number
`limit`	Number	Number of items per page
`total`	Number	Total number of items across all pages
`pages`	Number	Total number of pages

Pagination Best Practices

1. **Default Values**: Use default values (page=1, limit=25) for initial requests
2. **Limit Considerations**: Keep limits within the supported range (25–100) to avoid performance issues
3. **Page Navigation**: Use the `pages` field to determine total available pages
4. **Consistent Limits**: Use the same limit value when navigating through pages
5. **Error Handling**: Handle cases where requested page exceeds available pages

Supported Endpoints

The following endpoints support pagination:

- `GET /api/v1/resource/resources` - Retrieve all resources
- `GET /api/v1/resource/tags` - Retrieve all tags
- `GET /api/v1/resource/groups` - Retrieve all resource groups
- `GET /api/v1/resource/txStreamInfos` - Retrieve all TX stream infos
- `GET /api/v1/av-matrix/routes` - Retrieve all AV matrix routes
- `GET /api/v1/av-matrix/presets` - Retrieve all AV matrix presets
- `GET /api/v1/video-wall/video-walls` - Retrieve all video walls

API Versioning

The current version is `v1` (`/api/v1/`) with the latest release being `1.1.0`. Future versions will be available at `/api/v2/`, etc. Use the `GET /api` endpoint to discover supported versions and deprecation schedules.

Change Log

Version	Date	Changes
1.1.0	2025-08-08	Added URL decoding, pagination, RS232 presets, advanced resource filtering, specialized route creation, stream type optimization, and HDCP configuration endpoints

Version	Date	Changes
1.0.0	2025-07-24	Initial release

Examples

This section provides practical examples for common API operations. Examples are organized by functionality and include multiple programming languages where applicable.

Authentication Examples

Login (cURL)

```
curl -X POST https://<your-server>/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"user":"admin","password":"proav101"}'
```

Response:

```
{
  "success": true,
  "data": {
    "sessionId": "uuid-session-id"
  },
  "message": "Success",
  "timestamp": "2024-01-01T12:00:00.000Z"
}
```

Logout (cURL)

```
# Logout with session ID in request body
curl -X POST https://<your-server>/api/v1/auth/logout \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{"sessionValue":"uuid-session-id"}'

# Logout using cookie (no session ID needed)
curl -X POST https://<your-server>/api/v1/auth/logout \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json"
```

Validate Session (cURL)

```
curl -X POST https://<your-server>/api/v1/auth/validate \
  -H "Content-Type: application/json" \
  -d '{"sessionValue":"uuid-session-id"}'
```

API Discovery Examples

Get API Information (cURL)

```
curl -X GET https://<your-server>/api \
  -H "Content-Type: application/json"
```

Resource Management Examples

Get All Resources (cURL)

```
# Basic request
curl -X GET "https://<your-server>/api/v1/resource/resources" \
  -H "Cookie: auth_session=uuid-session-id"

# With pagination
curl -X GET "https://<your-server>/api/v1/resource/resources?page=1&limit=25" \
  -H "Cookie: auth_session=uuid-session-id"

# With filters
curl -X GET "https://<your-server>/api/v1/resource/resources?
deviceType=ENCODER&groupName=production" \
  -H "Cookie: auth_session=uuid-session-id"

# With keywords search
curl -X GET "https://<your-server>/api/v1/resource/resources?
keywords=encoder&tagName=production" \
  -H "Cookie: auth_session=uuid-session-id"

# With pagination and filters
curl -X GET "https://<your-server>/api/v1/resource/resources?
page=1&limit=50&deviceType=ENCODER&groupName=production" \
  -H "Cookie: auth_session=uuid-session-id"
```

Get Single Resource (cURL)

```
curl -X GET "https://<your-server>/api/v1/resource/resource?resourceName=encoder-
01" \
  -H "Cookie: auth_session=uuid-session-id"
```

Get TX Stream Infos (cURL)

```
# Basic request
curl -X GET "https://<your-server>/api/v1/resource/txStreamInfos" \
  -H "Cookie: auth_session=uuid-session-id"

# With pagination
curl -X GET "https://<your-server>/api/v1/resource/txStreamInfos?page=1&limit=50"
\
  -H "Cookie: auth_session=uuid-session-id"
```

Get TX Stream Info (cURL)

```
curl -X GET "https://<your-server>/api/v1/resource/txStreamInfo?
resourceName=encoder-01" \
  -H "Cookie: auth_session=uuid-session-id"
```

Select RX Input (cURL)

```
curl -X POST "https://<your-server>/api/v1/resource/rx/inputSelection?
resourceName=decoder-01&type=video" \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{"inputSelection":"IP"}'
```

Select TX Input (cURL)

```
curl -X POST "https://<your-server>/api/v1/resource/tx/inputSelection?
resourceName=encoder-01&type=video" \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{"inputSelection":"HDMI"}'
```

Get RS232 Presets (cURL)

```
curl -X GET "https://<your-server>/api/v1/resource/rs232/presets?
resourceName=decoder-01" \
  -H "Cookie: auth_session=uuid-session-id"
```

Send RS232 Preset Command (cURL)

```
curl -X POST "https://<your-server>/api/v1/resource/rs232/send?
presetName=power_on&resourceName=decoder-01" \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json"
```

Configure HDCP Input Settings (cURL)

```
curl -X POST "https://<your-server>/api/v1/resource/hdcp/input?
resourceName=encoder-01" \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{"hdcpCapability":"AUTO"}'
```

Configure HDCP Output Settings (cURL)

```
curl -X POST "https://<your-server>/api/v1/resource/hdcp/output?
resourceName=encoder-01" \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{"hdcpType":"HDCP_FOLLOW_INPUT"}'
```

AV Matrix Management Examples

Get Routes (cURL)

```
curl -X GET "https://<your-server>/api/v1/av-matrix/routes?page=1&limit=25" \
  -H "Cookie: auth_session=uuid-session-id"
```

Get Presets (cURL)

```
curl -X GET "https://<your-server>/api/v1/av-matrix/presets?page=1&limit=25" \
  -H "Cookie: auth_session=uuid-session-id"
```

Create Route (cURL)

```

```

```
curl -X POST https://<your-server>/api/v1/av-matrix/route \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1",
    "videoType": "HC",
    "audioType": "LC",
    "usb": true,
    "rs232": false
  }'
```

Create Data Route (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/dataRoute \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1"
  }'
```

Create AV Route (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/avRoute \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1",
    "streamType": "HC"
  }'
```

Create KVM Route (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/kvmRoute \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1",
    "streamType": "HC"
  }'
```

Delete Route (cURL)

```
curl -X DELETE https://<your-server>/api/v1/av-matrix/route \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1",
    "videoType": "HC",
    "audioType": "LC",
    "usb": true,
    "rs232": false
  }'
```

Delete Data Route (cURL)

```
curl -X DELETE https://<your-server>/api/v1/av-matrix/dataRoute \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1"
  }'
```

Delete AV Route (cURL)

```
curl -X DELETE https://<your-server>/api/v1/av-matrix/avRoute \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1",
    "streamType": "HC"
  }'
```

Delete KVM Route (cURL)

```
curl -X DELETE https://<your-server>/api/v1/av-matrix/kvmRoute \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": "receiver1",
    "txName": "transmitter1",
```

```
    "streamType": "HC"
  }'
```

Route to All (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/routeToAll \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": ["receiver1", "receiver2", "receiver3"],
    "txName": "transmitter1",
    "videoType": "HC",
    "audioType": "LC",
    "rs232": true
  }'
```

Route to All Data (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/routeToAllData \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": ["receiver1", "receiver2", "receiver3"],
    "txName": "transmitter1"
  }'
```

Route to All AV (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/routeToAllAv \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
    "rxName": ["receiver1", "receiver2", "receiver3"],
    "txName": "transmitter1",
    "streamType": "HC"
  }'
```

Save Preset (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/preset/save \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json" \
  -d '{
```

```
    "presetName": "my_preset_001",
    "userName": "admin"
  }'
```

Apply Preset (cURL)

```
curl -X POST https://<your-server>/api/v1/av-matrix/preset/apply/my_preset_001 \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json"
```

Video Wall Management Examples

Get Video Walls (cURL)

```
curl -X GET "https://<your-server>/api/v1/video-wall/video-walls?page=1&limit=25"
\
  -H "Cookie: auth_session=uuid-session-id"
```

Set Video Wall State (cURL)

```
curl -X POST "https://<your-server>/api/v1/video-wall/setState?
videoWallName=wall1&state=enable" \
  -H "Cookie: auth_session=uuid-session-id" \
  -H "Content-Type: application/json"
```

URL Encoding Examples

Resource Search with Special Characters (cURL)

```
# Search for resources with "&" in the name
curl -X GET "https://<your-server>/api/v1/resource/resources?
name=Video%26Audio%20System" \
  -H "Cookie: auth_session=uuid-session-id"

# Search for resources with spaces and special characters
curl -X GET "https://<your-server>/api/v1/resource/resources?
type=HD%2B4K%20Display&location=Main%20Room%20%231" \
  -H "Cookie: auth_session=uuid-session-id"
```

Python Code Examples

Authentication Functions

```python
import requests

def login(base_url, username, password):
    """Login and obtain session ID"""
    url = f"{base_url}/api/v1/auth/login"
    payload = {"user": username, "password": password}
    headers = {"Content-Type": "application/json"}

    response = requests.post(url, json=payload, headers=headers)
    return response.json()

def logout(base_url, session_id):
    """Logout and invalidate session"""
    url = f"{base_url}/api/v1/auth/logout"
    headers = {"Cookie": f"auth_session={session_id}"}
    data = {"sessionValue": session_id}

    response = requests.post(url, json=data, headers=headers)
    return response.json()

def logout_with_cookie(base_url):
    """Logout using cookie (no session ID needed)"""
    url = f"{base_url}/api/v1/auth/logout"
    headers = {"Content-Type": "application/json"}

    response = requests.post(url, headers=headers)
    return response.json()

# Usage
base_url = "https://your-server"
session_data = login(base_url, "admin", "proav101")
if session_data["success"]:
    session_id = session_data["data"]["sessionId"]
    print(f"Login successful: {session_id}")

    # Use session for other operations
    logout_result = logout(base_url, session_id)
    print(f"Logout result: {logout_result}")
else:
    print(f"Login failed: {session_data['message']}")

# Or logout with cookie
result = logout_with_cookie()
if result["success"]:
    print("Logout successful")
```

Resource Management Functions

```python
import requests

def get_resources(base_url, session_id, page=1, limit=25, **filters):
    """Get resources with optional filtering and pagination"""
    url = f"{base_url}/api/v1/resource/resources"
    headers = {"Cookie": f"auth_session={session_id}"}
    params = {"page": page, "limit": limit, **filters}

    response = requests.get(url, headers=headers, params=params)
    return response.json()

def get_device_status(base_url, session_id, resource_name):
    """Get status of a specific device"""
    url = f"{base_url}/api/v1/resource/resource"
    headers = {"Cookie": f"auth_session={session_id}"}
    params = {"resourceName": resource_name}

    response = requests.get(url, headers=headers, params=params)
    return response.json()

# Usage examples
base_url = "https://your-server"
session_id = "uuid-session-id"

# Get all resources with pagination
resources = get_resources(base_url, session_id, page=1, limit=10)
if resources["success"]:
    print(f"Found {len(resources['data'])} resources")

# Get resources filtered by device type
encoders = get_resources(base_url, session_id, deviceType="ENCODER")
if encoders["success"]:
    print(f"Found {len(encoders['data'])} encoders")

# Get specific device status
device_status = get_device_status(base_url, session_id, "encoder-01")
if device_status["success"]:
    print(f"Device state: {device_status['data']['deviceInfo']['state']}")
```

AV Matrix Route Creation

```python
import requests

def create_av_route(base_url, session_id, rx_name, tx_name, stream_type="HC"):
    """Create an AV route between receiver and transmitter"""
    url = f"{base_url}/api/v1/av-matrix/avRoute"
    headers = {
        "Cookie": f"auth_session={session_id}",
        "Content-Type": "application/json"
    }
```

```
    data = {
        "rxName": rx_name,
        "txName": tx_name,
        "streamType": stream_type
    }

    response = requests.post(url, json=data, headers=headers)
    return response.json()

def create_kvm_route(base_url, session_id, rx_name, tx_name, stream_type="HC"):
    """Create a KVM route with video, audio, and USB"""
    url = f"{base_url}/api/v1/av-matrix/kvmRoute"
    headers = {
        "Cookie": f"auth_session={session_id}",
        "Content-Type": "application/json"
    }
    data = {
        "rxName": rx_name,
        "txName": tx_name,
        "streamType": stream_type
    }

    response = requests.post(url, json=data, headers=headers)
    return response.json()

# Usage
result = create_av_route(base_url, session_id, "receiver1", "transmitter1", "HC")
if result["success"]:
    print(f"AV route created: {result['data']['createdLinkTypes']}")

kvm_result = create_kvm_route(base_url, session_id, "receiver1", "transmitter1",
"LC")
if kvm_result["success"]:
    print(f"KVM route created: {kvm_result['data']['createdLinkTypes']}")
```

JavaScript Code Examples

Authentication Functions

```
// Login and obtain session ID
async function login(username, password) {
    try {
        const response = await fetch('/api/v1/auth/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ user: username, password: password })
        });
```

```
        const data = await response.json();
        if (data.success) {
            return data.data.sessionId;
        } else {
            throw new Error(data.message);
        }
    } catch (error) {
        console.error('Login failed:', error.message);
        throw error;
    }
}

// Logout and invalidate session
async function logout(sessionId) {
    try {
        const response = await fetch('/api/v1/auth/logout', {
            method: 'POST',
            headers: {
                'Cookie': `auth_session=${sessionId}`,
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ sessionValue: sessionId })
        });

        const data = await response.json();
        return data.success;
    } catch (error) {
        console.error('Logout failed:', error.message);
        throw error;
    }
}

// Logout using cookie (no session ID needed)
async function logoutWithCookie() {
    try {
        const response = await fetch('/api/v1/auth/logout', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            }
        });

        const data = await response.json();
        return data.success;
    } catch (error) {
        console.error('Logout failed:', error.message);
        throw error;
    }
}

// Usage
async function authenticate() {
    try {
        const sessionId = await login('admin', 'proav101');
```

```
        console.log('Login successful:', sessionId);

        // Use session for other operations
        const logoutSuccess = await logout(sessionId);
        console.log('Logout successful:', logoutSuccess);
    } catch (error) {
        console.error('Authentication failed:', error.message);
    }
}

authenticate();
```

Resource Management Functions

```
// Get resources with optional filtering and pagination
async function getResources(sessionId, page = 1, limit = 25, filters = {}) {
    try {
        const params = new URLSearchParams({
            page: page.toString(),
            limit: limit.toString(),
            ...filters
        });

        const response = await fetch(`/api/v1/resource/resources?${params}`, {
            headers: {
                'Cookie': `auth_session=${sessionId}`
            }
        });

        const data = await response.json();
        return data;
    } catch (error) {
        console.error('Failed to get resources:', error.message);
        throw error;
    }
}

// Get status of a specific device
async function getDeviceStatus(sessionId, resourceName) {
    try {
        const response = await fetch(`/api/v1/resource/resource?
resourceName=${encodeURIComponent(resourceName)}`, {
            headers: {
                'Cookie': `auth_session=${sessionId}`
            }
        });

        const data = await response.json();
        return data;
    } catch (error) {
        console.error('Failed to get device status:', error.message);
```

```
        throw error;
    }
}

// Usage examples
async function manageResources(sessionId) {
    try {
        // Get all resources with pagination
        const resources = await getResources(sessionId, 1, 10);
        if (resources.success) {
            console.log(`Found ${resources.data.length} resources`);
        }

        // Get resources filtered by device type
        const encoders = await getResources(sessionId, 1, 25, { deviceType:
'ENCODER' });
        if (encoders.success) {
            console.log(`Found ${encoders.data.length} encoders`);
        }

        // Get specific device status
        const deviceStatus = await getDeviceStatus(sessionId, 'encoder-01');
        if (deviceStatus.success) {
            console.log(`Device state: ${deviceStatus.data.deviceInfo.state}`);
        }
    } catch (error) {
        console.error('Resource management failed:', error.message);
    }
}
```

AV Matrix Route Creation

```
// Create an AV route between receiver and transmitter
async function createAvRoute(sessionId, rxName, txName, streamType = 'HC') {
    try {
        const response = await fetch('/api/v1/av-matrix/avRoute', {
            method: 'POST',
            headers: {
                'Cookie': `auth_session=${sessionId}`,
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                rxName,
                txName,
                streamType
            })
        });

        const data = await response.json();
        return data;
    } catch (error) {
```

```
        console.error('Failed to create AV route:', error.message);
        throw error;
    }
}

// Create a KVM route with video, audio, audio, and USB
async function createKvmRoute(sessionId, rxName, txName, streamType = 'HC') {
    try {
        const response = await fetch('/api/v1/av-matrix/kvmRoute', {
            method: 'POST',
            headers: {
                'Cookie': `auth_session=${sessionId}`,
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                rxName,
                txName,
                streamType
            })
        });

        const data = await response.json();
        return data;
    } catch (error) {
        console.error('Failed to create KVM route:', error.message);
        throw error;
    }
}

// Usage
async function createRoutes(sessionId) {
    try {
        // Create AV route
        const avResult = await createAvRoute(sessionId, 'receiver1',
'transmitter1', 'HC');
        if (avResult.success) {
            console.log('AV route created:', avResult.data.createdLinkTypes);
        }

        // Create KVM route
        const kvmResult = await createKvmRoute(sessionId, 'receiver1',
'transmitter1', 'LC');
        if (kvmResult.success) {
            console.log('KVM route created:', kvmResult.data.createdLinkTypes);
        }
    } catch (error) {
        console.error('Route creation failed:', error.message);
    }
}
```

Additional Resources

- **API Discovery**: Use `GET /api` to explore available endpoints and features
- **Error Handling**: Check response status codes and error messages
- **Rate Limiting**: Respect API rate limits for optimal performance
- **Best Practices**: Follow security guidelines and use HTTPS in production

---

Support

For assistance, contact the PlexusAV support team at sales@plexusav.com or call +1.605.378.4800. Refer to the internal documentation at `/api/docs` for additional resources.