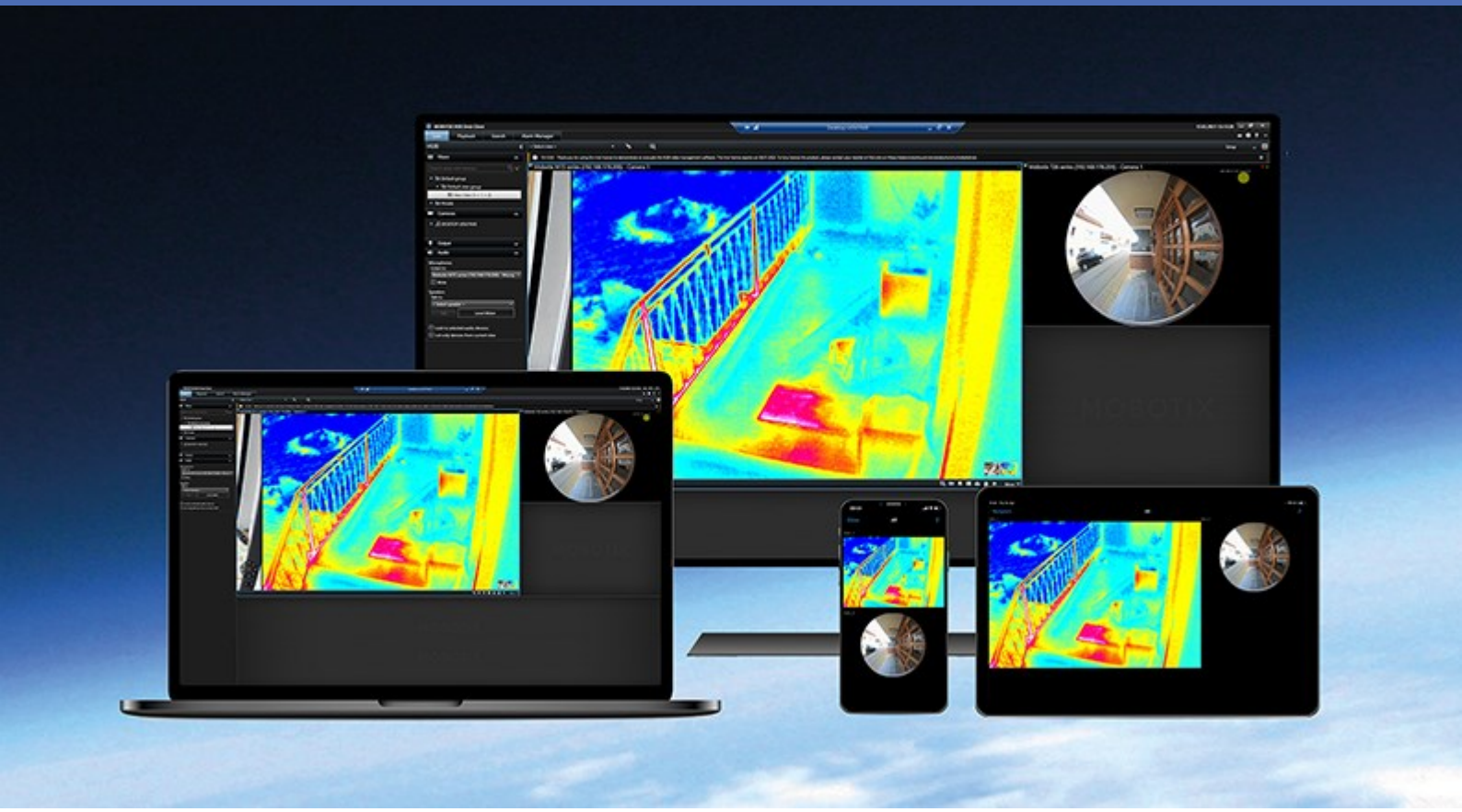
Administrator manual

MOBOTIX HUB API Gateway 2023 R1



BeyondHumanVision

MOBOTIX

Contents

Copyright

MOBOTIX AG • Kaiserstrasse • D-67722 Langmeil • Tel.: +49 6302 9816 0 • sales@mobotix.com • www.mobotix.com

Overview

Introduction

The MOBOTIX HUB VMS is planned to include RESTful APIs and streaming protocols that expose the functionality currently available through native .NET libraries and various proprietary protocols.

The API Gateway is installed on-premise and is intended to serve as a front-end and common entry point for RESTful API and streaming services on all the current VMS server components (management server, event server, recording servers, log server, etc). An API Gateway can be installed on the same host as the management server or separately, and more than one can be installed (each on their own host).

What's new?

MOBOTIX HUB VMS 2023 R1

- Some changes to API Gateway configuration files.
- WebRTC is enabled by default and now supports mDNS.

MOBOTIX HUB VMS 2022 R3

- The API Gateway can be configured to support Cross-Origin Resource Sharing (CORS).
- Pre-release of WebRTC support.

MOBOTIX HUB VMS 2022 R2

- A number of syntax issues in the OpenAPI spec file have been fixed.
- More complete coverage of the Configuration API.
- **Breaking change**: In some parts of the RESTful API, booleans were treated as strings, meaning that the values when provided as input would have to be enclosed in quotation marks and also would be returned in this form. As this is not in compliance with the JSON standard, and also not in accordance the OpenAPI spec file we provide, we have decided to change it to use true/false without the quotation marks.

Limitations

- Only the Configuration API is exposed as a RESTful API through the API Gateway.
- WebRTC cannot create a connection through a symmetric NAT firewall without using a TURN (Traversal Using Relays around NAT) server. Currently, the API Gateway does not support using a TURN server.
- To upgrade from the 2021 R2 pre-release of the API Gateway to later releases, you'll have to uninstall the 2021 R2 pre-release before upgrading.

Intended audience

This document is primarily aimed at system integrators and IT administrators. You are assumed to be somewhat familiar with MOBOTIX HUB VMS products.

API Gateway features

The MOBOTIX HUB VMS offers a number of APIs to support integrations. The full functionality is currently available through a plug-in environment, through native .NET libraries, and through various SOAP and native protocols. These APIs are used internally by MOBOTIX HUB VMS, and a large number of integrations have been developed using these APIs. But they are not practical for integrations in a cloud environment:

- The SOAP-based protocols relies on Windows Communication Framework (WCF) which is part of .NET Framework, making it difficult to implement non-Windows integrations.
- Media data streaming uses a proprietary protocol.
- To use the protocols, your integration must keep track of a number of service endpoints.

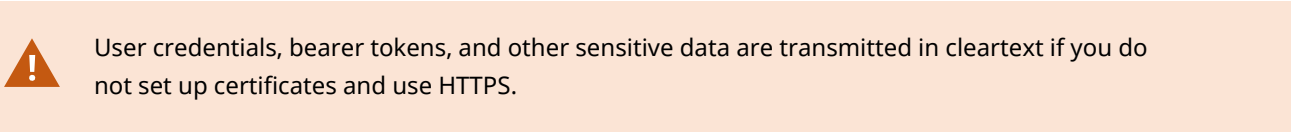
The API Gateway simplifies this by providing a single entry point for all services.

Authentication and authorization

The API Gateway relies on an OpenID Connect and OAuth 2.0 Identity Provider (IDP) for authentication and authorization.

To use the API Gateway, a client first authenticates and requests an access token from the Identity Provider. The client receives a bearer token that grants privileges to access services and to perform operations, as determined by the user's roles.

The client now uses the bearer token in the authorization header in subsequent requests. The client renews the bearer token before it expires by posting a new access token request with the same credentials.



RESTful APIs

The API Gateway acts as broker, routing requests and responses between external clients and the various downstream MOBOTIX HUB VMS services.

The RESTful API is implemented in part by each specific VMS server component, and the API Gateway can simply pass-through these requests and responses, while for other requests, the API Gateway will convert requests and responses as appropriate.

WebRTC

WebRTC is a peer-to-peer real-time communication framework, for example for video media data, based on open protocols (RTP, RTCP, and SCTP). WebRTC is attractive for cloud-based services because:

- most modern web browsers support WebRTC, eliminating the need for installing plug-ins,
- in many cases, media traffic can be routed directly between the peers, reducing the need for intermediary servers.

The API Gateway supports:

- a WebRTC signaling server that offers a simple RESTful API for establishing WebRTC connections,
- streaming live H.264 encoded video from a camera installed on a recording server through the WebRTC connection.

WebRTC cannot create a connection through a symmetric NAT firewall without using a TURN (Traversal Using Relays around NAT) server. Currently, the API Gateway does not support using a TURN server.

Requirements and considerations

Considerations

For development, set up separate development system

You are recommended to use a separate development system.

Use HTTPS

You should consider setting up a server certificate and using HTTPS. While the IDP, API Gateway, and the Management Server all can work with either HTTP or HTTPS, production systems should be set up with server certificates.

> ⚠ User credentials, bearer tokens, and other sensitive data are transmitted in cleartext if you do not set up certificates and use HTTPS.

Requirements

Installation of the HUB API Gateway requires MOBOTIX HUB VMS 2022 R1 or later

Server certificates and host names

If you set up the management server with encryption, you must also set up all API Gateway instances that connect to the management server with encryption. To enable this, the IIS on the host that you install the API Gateway on must be set up with a server certificate.

The server hostname you specify during installation of the API Gateway is used to connect the API Gateway to the identity provider service (IDP) and management server in the MOBOTIX HUB VMS, and should match a DNS name in the management server certificate.

MOBOTIX HUB users

The API Gateway installer must be able to log in to the MOBOTIX HUB VMS during the installation. The Windows user account that you used for installing the MOBOTIX HUB VMS has been added in the MOBOTIX HUB VMS to the Administrators role. You can use the same Windows account when you install the API Gateway.

To authenticate and access the API Gateway, you need an MOBOTIX HUB Basic user account.

- You can create an MOBOTIX HUB Basic user during installation of the MOBOTIX HUB VMS.
- After installation, you can use the **MOBOTIX HUB Management Client** to create MOBOTIX HUB Basic or Windows users.

WebRTC

Cross-Origin Resource Sharing CORS

Yoy need to enable CORS if the sample webpage is not served from the same origin host URL as the API Gateway, see Cross-Origin Resource Sharing (CORS) on page 23.

No WebRTC connection through a symmetric NAT firewall

WebRTC cannot create a connection through a symmetric NAT firewall without using a TURN (Traversal Using Relays around NAT) server. Currently, the API Gateway does not support using a TURN server.

Check with your system administrator if you are behind a symmetric NAT firewall, or run the test described here: Am I behind a Symmetric NAT?[1].

In that case, a WebRTC connection is possible only within the symmetric NAT firewall, that is, both your browser and the API Gateway must be behind the firewall.

WebRTC connection on a local network uses mDNS

To prevent private IP addresses from leaking from a local network when running WebRTC applications, modern browsers by default send mDNS (multicast DNS) addresses as ICE Candidates to the signaling server.

API Gateway support for mDNS

The signaling server running in the API Gateway supports resolving mDNS addresses when running on a Windows version with native support for mDNS.

Native support for mDNS was introduced in Windows version 1809 (October 2018) or later, and is available in any recently updated Windows Server 2019 or Windows 10 installations, and all Windows Server 2022 and Windows 11 installations.

WebRTC connections across routers in a local network

mDNS relies on multicast which by default will not pass through routers. This means that in enterprise environments, mDNS will fail in many cases:

- mDNS over wired Ethernet works on the same local network segment, but in more complex network solution (most enterprise environments), mDNS will fail.
- mDNS over WiFi will only work on simple network configurations (as for wired networks). In configurations with WiFi extender or Mesh networks, mDNS will likely fail.

The signaling server running in the API Gateway supports a workaround for connections across routers on a local network. The signaling server will attempt to get the client's local IP network address from `X-Forwarded-For` and

[1] https://webrtchacks.com/symmetric-nat/

`Remote_Addr` headers in the HTTP request and use that to add an ICE Candidate with higher priority than the ICE Candidate with the mDNS address. This will not work in all cases; on some networks, `X-Forwarded-For` is removed and `Remote_Addr` will not contain the local IP address of client.

Installation

Installation overview

This installation overview describes the following tasks:

- Installing the API Gateway, either during or after the initial installation of the MOBOTIX HUB VMS.
- Verifying that an API Gateway is operational.

Install the API Gateway

There are several ways to install an API Gateway:

- During installation of a either a **Single computer** or a **Custom** (distributed) MOBOTIX HUB VMS, using the MOBOTIX HUB VMS product installer.
- After installation of a MOBOTIX HUB VMS, using the API Gateway installer downloaded from the management server's Administrative Installation Page.

Install the API Gateway using the MOBOTIX HUB VMS product installer

The API Gateway is included by default in both **Single computer** and **Custom** installations.

The API Gateway will be installed on the same host as the management server and with the same server certificate if you enable encryption.

Install the API Gateway using the API Gateway installer

The management server has a built-in public installation web page. From this web page, administrators and end-users can download and install additional system components. For example, if you didn't initially install an API Gateway, you can install it later.

1. From the computer where management server is installed, go to the management server's download web page. In Windows' **Start** menu, select **MOBOTIX** > **Administrative Installation Page** and write down or copy the address for later use when installing the system components on the other computers. The address is typically `http://[management server address]/installation/Admin/default-en-US.htm`.
2. Log into the computer where you want to install the API Gateway.
3. Open the management server's download web page in a web browser.
4. Locate the **API Gateway Installer** section, and select **All Languages** to start downloading the installer. Save the installer first, or run it directly from the web page.
5. Choose installation language.
6. Accept license terms.
7. Enter the management server address. Use HTTPS if the management server is configured to be secure.

8. Select the web site on the local IIS to use with the API Gateway, usually the Default Web Site.
9. If the management server is configured to be secure, you must select a server certificate for the API Gateway host. If you are installing the API Gateway on the host of the management server, select the server certificate you used when installing MOBOTIX HUB VMS.
10. Select the service account for the API Gateway, usually the Network Service account.
11. Select file location and product language.
12. Select **Install**.

Verify that the API Gateway is operational

Replace the hostname `test-01.example.com`, username `seamrune`, and password `Rad23Swops#` in the following request samples.
In Windows Command Prompt (CMD), replace the line continuation character `\` with `^`.

1. Verify that you can get a list of well-known URIs from the API Gateway:

cURL

```
curl --insecure --request GET "https://test-01.example.com/api/.well-known/uris"
```

PowerShell

```
$response = Invoke-RestMethod 'https://test-01.example.com/api/.well-known/uris' -Method 'GET'
$response | ConvertTo-Json
```

Response body

```
{
  "ProductVersion": "22.1.5804.1",
  "UnsecureManagementServer": "http://test-01.example.com/",
  "SecureManagementServer": "https://test-01.example.com/",
  "IdentityProvider": "https://test-01.example.com/IDP",
  "ApiGateways": [
    "https://test-01.example.com/API/"
  ]
}
```

In case you had installed an API Gateway on another host, you could use the hostname of that host.

2. Verify that you can authenticate and retrieve a bearer token from the built-in IDP.

cURL

```
curl --insecure --request POST "https://test-01.example.com/idp/connect/token" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode "grant_type=password" \
--data-urlencode "username=seamrune" \
--data-urlencode "password=Rad23Swops#" \
--data-urlencode "client_id=GrantValidatorClient"
```

PowerShell

```
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/x-www-form-urlencoded")
$body = @{grant_type='password'
    username='seamrune'
    password='Rad23Swops#'
    client_id='GrantValidatorClient'}
$response = Invoke-RestMethod 'https://test-01.example.com/idp/connect/token' `
    -Method 'POST' -Headers $headers -Body $body
$response | ConvertTo-Json
```

Response body

```
{
  "access_token": "eyJhbG . . . YTWPjg",
  "expires_in": 3600,
  "token_type": "Bearer",
  "scope": "managementserver"
}
```

Copy the `access_token` value from the response body; you will use the value as the bearer token value in the following request.

3. Verify that you can submit a request through the API Gateway.

   Replace the hostname `test-01.example.com` and the bearer token value `eyJhbG . . . YTWPjg` in the following request samples.

cURL

```
curl --insecure --request GET "https://test-01.example.com/api/rest/v1/sites" \
--header "Authorization: Bearer eyJhbG . . . YTWPjg"
```

PowerShell

```
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Authorization", "Bearer eyJhbG . . . YTWPjg")
$response = Invoke-RestMethod 'https://test-01.example.com/api/rest/v1/sites' `
    -Method 'GET' -Headers $headers
$response | ConvertTo-Json
```

Response body


```
{
  "array": [
    {
      "displayName": "TEST-01",
      "id": "2d12465c-3485-4ca8-a9fb-86a79de1a82f",
```

```
      "name": "TEST-01",
      "description": "",
      "lastModified": "2021-11-11T11:11:11.1111111Z",
      "timeZone": "Central Europe Time",
      "computerName": "TEST-01",
      "domainName": "example.com",
      "lastStatusHandshake": "2021-11-11T11:11:11.1111111Z",
      "physicalMemory": 0,
      "platform": "[Not Available]",
      "processors": 0,
      "serviceAccount": "S-1-5-20",
      "synchronizationStatus": 0,
      "masterSiteAddress": "",
      "version": "21.2.0.1",
      "relations": {
        "self": {
          "type": "sites",
          "id": "2d12465c-3485-4ca8-a9fb-86a79de1a82f"
        }
      }
    }
  ]
}
```

Configuration

API Gateway configuration files

API Gateway configuration files are located in the installation location, by default `%ProgramFiles%\MOBOTIX\HUB API Gateway\`.

These configuration files are relevant for the API Gateway:

- `appsettings.json`: Reverse proxy (routing), CORS, WebRTC, log levels, etc.
- `appsettings.Production.json`: Overrides the configuration settings in `appsettings.json`.
- `nlog.config`: Log layout, log targets, etc.

Editing configuration files

Use a validating editor to edit configuration files. Most popular code editors support JSON and XML syntax validation, either by default or through extensions.

appsettings.json and appsettings.Production.json

Do not edit `appsettings.json` manually. This file is created by the product installer and maintained by the Server Configurator.

If you need to override a configuration setting in `appsettings.json`, create `appsettings.Production.json` and add configuration overrides here. `appsettings.Production.json` will not be removed or changed by product updates.

To override configuration settings in `appsettings.json`, copy the complete top level property from `appsettings.json` to `appsettings.Production.json` and remove the nested properties that you don't want to change.

For example, to change the management server host address but no other `ReverseProxy` settings, include this in `appsettings.Production.json`:

```
1 {
2   "ReverseProxy": {
3     "Clusters": {
```

```
      "managementserver": {
        "Destinations": {
          "hostname": {
            "Address": "https://test-02.example.com/"
          }
        }
      }
    }
  }
}
```

If you add several properties to `appsettings.Production.json`, remember to include a comma between the properties, but no trailing comma:

```
{
  "Logging": {
    "LogLevel": {
      "Yarp": "Information"
    }
  },
  "ReverseProxy": {
    "Clusters": {
      "managementserver": {
        "Destinations": {
          "hostname": {
```

```
            "Address": "https://test-02.example.com/"
          }
        }
      }
    }
  }
}
```

Reverse proxy

The reverse proxy (routing) functionality of the API Gateway is implemented using **YARP**.

This part of `appsettings.json` is related to the reverse proxy functionality. The configuration is created by the product installer and maintained by the Server Configurator.

```
  "ReverseProxy": {
    "Routes": {
      "well-known": {
        "ClusterId": "managementserver",
        "Match": {
          "Path": "/.well-known/{**remainder}"
        },
        "Transforms": [
          {
            "PathPattern": "/ManagementServer/.well-known/{**remainder}"
          }
        ]
```

```
    },
    "rest-api": {
      "ClusterId": "managementserver",
      "Match": {
        "Path": "/rest/v1/{**remainder}"
      },
      "Transforms": [
        {
          "PathPattern": "/ManagementServer/Rest/{**remainder}"
        }
      ]
    },
    "idp": {
      "ClusterId": "managementserver",
      "Match": {
        "Path": "/IDP/{**remainder}"
      },
      "Transforms": [
        {
          "PathPattern": "/IDP/{**remainder}"
        }
      ]
    },
    "share": {
      "ClusterId": "managementserver",
      "Match": {
        "Path": "/share/{**remainder}"
```

```
        },
        "Transforms": [
          {
            "PathPattern": "/share/{**remainder}"
          },
          {
            "X-Forwarded": "Append"
          }
        ]
      }
    },
    "Clusters": {
      "managementserver": {
        "Destinations": {
          "hostname": {
            "Address": "https://test-01.example.com/"
          }
        }
      }
    }
  }
```

For more information about YARP, please refer to YARP: Yet Another Reverse Proxy. [1]

Cross-Origin Resource Sharing (CORS)

The API Gateway can be configured to support Cross-Origin Resource Sharing (CORS). The following response headers are supported:

[1] https://microsoft.github.io/reverse-proxy/index.html

- `Access-Control-Allow-Origin`[1]
- `Access-Control-Allow-Headers`[2]
- `Access-Control-Allow-Methods`[3]

CORS is disabled by default. You enable and configure CORS support by creating and editing `appsettings.Production.json`.

1. Create `appsettings.Production.json` (if not already created).
2. Enable and configure CORS response headers similar to this:

```
{
  "CORS": {
    "Enabled": true,
    "Access-Control-Allow-Origin": "yourdomain1.com,yourdomain2.com",
    "Access-Control-Allow-Headers": "Content-Type",
    "Access-Control-Allow-Methods": "*"
  }
}
```

3. Restart the IIS, or at least recycle `VideoOS ApiGateway AppPool`.

Only required response headers should be defined. Each response header can have multiple values, provided as a list of comma-separated values.

In a production system , always specify the `Access-Control-Allow-Origin` value with explicit origins. Never use wildcard (*) or null in your origin as this can put the security of your system at risk.

For development and test purposes, you can use a very permissive policy:

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin
[2] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Headers
[3] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Methods

```
{
  "CORS": {
    "Enabled": true,
    "Access-Control-Allow-Origin": "*",
    "Access-Control-Allow-Headers": "*",
    "Access-Control-Allow-Methods": "*"
  }
}
```

This will allow calls from any origin, including a local file system, to the API Gateway.

For more information about CORS, please refer to Cross-Origin Resource Sharing (CORS)[1].

WebRTC

WebRTC is a peer-to-peer protocol for streaming data, for example video.

STUN server address

To help establish a connection through NATs, WebRTC uses a STUN (Session Traversal Utilities for NAT) server.

By default, the API Gateway use the Google STUN server `stun1.l.google.com:19302`.

To specify another STUN server:

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

1. Create `appsettings.Production.json` (if not already created).
2. Add the configuration, for example:

```
{
  "WebRTC": {
    "STUN": {
      "Hostname": {
        "Address": "stun:stun.example.com:19302"
      }
    }
  }
}
```

3. Restart the IIS, or at least recycle `VideoOS ApiGateway AppPool`.

For more information about WebRTC, please refer to WebRTC API[1].

Logging

The API Gateway uses **NLog** for logging.

Logging is configured in two places:

- `appsettings.json` and `appsettings.Production.json`: Log levels
- `nlog.config`: Log layout, log targets, etc.

Log levels

This part of `appsettings.json` is related to logging:

```
  "Logging": {
```

[1] https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

```
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information",
      "Yarp": "Warning"
    }
  }
```

To include YARP routing log messages, add a log level setting in `appsettings.Production.json` for `Yarp`, for example, `Information`:

1. Create `appsettings.Production.json` (if not already created).
2. Add the configuration that you want to override, for example:

```
{
  "Logging": {
    "LogLevel": {
      "Yarp": "Information"
    }
  }
}
```

3. Restart the IIS, or at least recycle `VideoOS ApiGateway AppPool`.

Log layout, log targets, etc

This is the default NLog configuration file `nlog.config`:

Configuration

```
<?xml version="1.0" encoding="utf-8"?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  autoReload="true"
  internalLogLevel="Warn"
  internalLogFile="internal-nlog.txt">
  <variable
    name="logDirectory"
     value="C:\ProgramData\MOBOTIX\ApiGateway\Logs" />
  <variable
    name="archiveDirectory"
   value="${var:logDirectory}\Archive" />
  <variable
    name="defaultLayout"
    value="${date:format=yyyy-MM-dd HH\:mm\:ss.fffzzz} [${threadid:padding=6}]
${level:uppercase=true:padding=-10} - ${message} ${exception:format=tostring}" />
  <targets>
    <target
      name="logfile"
      xsi:type="File"
      fileName="${var:logDirectory}\gateway.log"
      archiveFileName="${var:archiveDirectory}\gateway-{####}.log"
      archiveNumbering="Rolling"
      maxArchiveFiles="20"
      archiveEvery="Day"
      archiveAboveSize="1000000"
      archiveOldFileOnStartup="true"
```

```
27        createDirs="true"
28        layout="${var:defaultLayout}" />
29    </targets>
30    <rules>
31      <logger name="*" minlevel="Debug" writeTo="logfile" />
32    </rules>
33  </nlog>
```

With the configuration setting `autoReload="true"`, NLog will monitor and reload the configuration file whenever it is modified. You can change the log configuration on the fly without restarting anything.

For more information about NLog configuration, please refer to NLog Configuration options.[1]

[1] https://nlog-project.org/config

Troubleshooting

CORS errors

You attempt to log in and create a WebRTC connection in a browser application to the API Gateway, but the requests from the application are blocked by the browser.

CORS error symptoms

Browser-based applications, for example WebRTC applications, usually fetch resources from various origins. For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts.

For example, a web page with JavaScript code loads from one origin URL, and the JavaScript code attempts to fetch some resources from another origin URL. The browser blocks access to those resources unless they are served with CORS headers.

IIS configuration issues might also appear as CORS errors.

In your browser Developer tools **Console** tab, you will see errors similar to these:

Developer tool | Console tab

```
1  Access to fetch at 'http://test-01/api/IDP/connect/token' from origin 'http://localhost'
   has been blocked by CORS policy: . . .

2  Access to fetch at 'http://test-01/api/REST/v1/WebRTC/Session' from origin
   'http://localhost' has been blocked by CORS policy: . . .
```

Cause

The webpage is not served from same host server URL as the API Gateway, and CORS support has not been enabled.

Remedy

Enable CORS support as described in Cross-Origin Resource Sharing (CORS) on page 23.

Cause

Errors are sometime presented in the browser as CORS error without being actual CORS issues. If you see a CORS error message in the browser, it could be related to configuration issues in the IIS.

Remedy

Open your browser Developer tools and select the **Network** tab. If it is not an CORS error, the actual error will be shown here in the messages received before the CORS error.

No WebRTC connection

You attempt to log in and create a WebRTC connection in a browser application to the API Gateway. The log in succeeds, but the application fails to create a WebRTC connection.

WebRTC connection through a symmetric NAT firewall

WebRTC cannot create a connection through a symmetric NAT firewall without using a TURN (Traversal Using Relays around NAT) server. Currently, the API Gateway does not support using a TURN server.

Check with your system administrator if you are behind a symmetric NAT firewall, or run the test described here: Am I behind a Symmetric NAT?[1].

Remedy

Currently, there is no remedy.

WebRTC connection on a local network uses mDNS

To prevent private IP addresses from leaking from a local network when running WebRTC applications, modern browsers by default send mDNS (multicast DNS) addresses as ICE Candidates to the signaling server.

mDNS relies on multicast which by default will not pass through routers. This means that in enterprise environments, mDNS will fail in many cases.

The signaling server running in the API Gateway supports a workaround for connections across routers on a local network. The signaling server will attempt to get the client's local IP network address from `X-Forwarded-For` and `Remote_Addr` headers in the HTTP request and use that to add an ICE Candidate with higher priority than the ICE Candidate with the mDNS address. This will not work in all cases; on some networks, `X-Forwarded-For` is removed and `Remote_Addr` will not contain the local IP address of client.

Remedy

As a last resort, you can try disabling browser mDNS support to force the browser to reveal the local IP network address in WebRTC connections.

In Chromium-based browsers, mDNS support can be disabled by opening `chrome://flags` or `edge://flags` and setting **Anonymize local IPs exposed by WebRTC** to **Disabled**.

[1] https://webrtchacks.com/symmetric-nat/

API Gateway doesn't answer requests

You attempt to log in in a browser application through the API Gateway. The API Gateway doesn't answer requests and log in doesn't succeed.

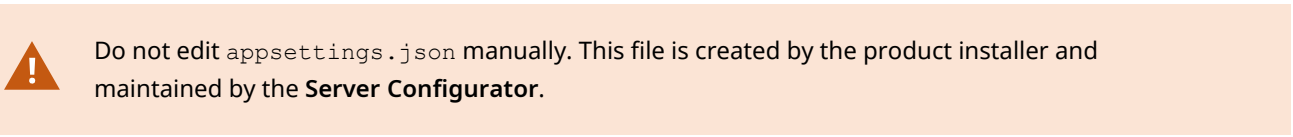
Symptoms

- Your browser Developer tools **Network** tab shows error status `502 Bad Gateway` or `503 Service Unavailable`.
- You see error events in the Windows Application log and IIS request log related to the API Gateway.

Cause

Syntax errors in the appsettings configuration files will prevent the API Gateway from starting.

Remedy



Do not edit `appsettings.json` manually. This file is created by the product installer and maintained by the **Server Configurator**.

Open and edit `appsettings.production.json` in a validating editor.

For more information, see Editing configuration files on page 19.

MOBOTIX

BeyondHumanVision

MOBOTIX AG • Kaiserstrasse • D-67722 Langmeil • Tel.: +49 6302 9816 0 • sales@mobotix.com • www.mobotix.com