GEUTEBRÜCK

G-Core Web API Dokumentation

Version: 2.0

04.01.2024

Inhaltsverzeichnis

Über diese Dokumentation

Aktuelle Softwareversion: **G-Core Web API** 2.0.

Rechtlicher Hinweis

Ohne vorherige Genehmigung darf diese Dokumentation weder vollständig noch in Auszügen kopiert, übersetzt oder in eine maschinenlesbare Form gebracht werden.

Die GEUTEBRÜCK GmbH übernimmt keine Gewähr für die Richtigkeit der Angaben in dieser Dokumentation sowie für die Software oder darin enthaltene Angaben. Jede konkludente Gewährleistung, Zusicherung marktgängiger Qualität oder Eignung für einen bestimmten Zweck hinsichtlich der Dokumentation, der Software und anderer Angaben wird hiermit ausdrücklich abgelehnt.

Die GEUTEBRÜCK GmbH haftet unter keinen Umständen für mittelbare oder unmittelbare Folgeschäden oder besondere Schadensfolgen, die sich aus oder in Verbindung mit dieser Dokumentation ergeben, gleichgültig, ob diese aufgrund unerlaubter Handlungen, eines Vertrages oder sonstigen Gründen in Verbindung mit dieser Dokumentation, der Software oder darin enthaltener oder verwendeter Angaben entstehen.

Die GEUTEBRÜCK GmbH behält sich das Recht vor, diese Dokumentation oder die darin enthaltenen Informationen jederzeit ohne Vorankündigung zu ändern. Die darin beschriebene Software unterliegt den Bedingungen eines gesonderten Lizenzvertrages.

Über die Web API

Die G-Core Web API ermöglicht die Verbindung zu G-Core über eine Reihe von plattformunabhängigen APIs. Die API wurde entwickelt, um einen einfachen Zugriff auf die Kernfunktionen von G-Core zu ermöglichen, wie z.B. Videostreams und Aktionen.

Die G-Core Web API ist besonders hilfreich, um webbasierte Dienste mit Ihrem G-Core VMS zu verbinden. Für native Windows-Anwendungen können Sie zwischen unserem C++/C#-basierten SDK und der G-Core Web API wählen.

Installation

Systemanforderungen

Für die Installation der Web API sind folgenden Systemanforderungen erforderlich:

- G-Core 7.0 oder neuer
- .NET 6.0 runtime oder neuer
- Lizenzen (siehe **Lizenzierung**)

Installation der Web API

Installieren Sie die Web API auf dem G-Core Server.

1. Führen Sie die Datei `G-Core_Web_API_installer_xxx.exe` aus.
2. Akzeptieren Sie die **Lizenzvereinbarung** und klicken Sie auf **Weiter**.
3. Klicken Sie im Dialogfenster **Ready to Install (Bereit zur Installation)** auf **Installieren**.
4. Wenn die Installation abgeschlossen ist, klicken Sie auf **Fertig stellen**.

Nach Abschluss der Installation wird ein laufender **G-Core Web API Service** in Ihren Windows-Diensten angezeigt.

Ein selbstsigniertes HTTPS-Zertifikat ist Teil der Installation. Für weitere Informationen siehe **HTTPS-Zertifikat**.

Lizenzierung

Die G-Core Web API ist ab G-Core Version 7.0 kostenlos enthalten. Die folgenden kostenlosen Lizenzen sind für die Nutzung der Web API erforderlich:

Lizenz	Beschreibung
8.35000 - Web API	Web API Lizenz für die Grundfunktion - 1x pro Server
8.35001 - Web API Metadaten	Web API Metadaten Lizenz für den Zugriff auf Metadaten (PLC-Endpunkt) - 1x pro Server
8.35002 - Web API ChannelConnect	Web API ChannelConnect Lizenz für Zugriff auf Streamingkanäle - 1x pro Kanal

Alle aktuellen und neuen Lizenzen sind mit den erforderlichen Lizenzen ausgestattet. Wenn Sie einen Dongle ohne diese Features haben, müssen Sie eine neue Lizenzdatei (*.lic oder *.slk) in Ihrem Lizenzportal erzeugen oder eine Lizenzdatei anfordern unter options@geutebrueck.com.

Upgrade 1.5.x auf 2.0

Während des Upgrades wird die vorhandene appsettings.json in appsettings-backup.json umbenannt. Es wird eine neue appsettings.json mit den Standardwerten erstellt.

Passen Sie Ihre Konfiguration entsprechend an. Normalerweise müssen nur die G-Core Zugangsdaten angepasst werden, wenn Sie die Web API nicht auf demselben System wie G-Core installiert haben.

HTTPS-Zertifikat

Für den Zugriff auf die Web API und ihre integrierte API-Dokumentation ist ein HTTPS-Zertifikat erforderlich.

Ein selbstsigniertes Zertifikat wird automatisch installiert und bei der Installation verwendet (Computerzertifikate). Alternativ kann auch ein Let's Encrypt-Zertifikat verwendet werden.

Verwendung von Let's Encrypt (ACME-Protokoll)

Let's Encrypt ermöglicht Ihnen die Verwaltung von Zertifikaten für eine bestimmte Domäne. Bestehende Zertifikate werden durch die Überprüfung des Ablaufdatums auf dem neuesten Stand gehalten und ein neues Zertifikat wird installiert, wenn keines vorhanden ist.

Wie Sie Let's Encrypt verwenden:

1. Installieren Sie die Web API.
2. Konfigurieren Sie Ihr Netzwerk (siehe **Netzwerk-Konfiguration**).
3. Bearbeiten Sie den Abschnitt `Let's Encrypt` in der Datei appsettings.json (siehe **Let's Encrypt Konfiguration**).
4. Starten Sie den Dienst neu.
5. Stellen Sie sicher, dass das Let's Encrypt-Zertifikat installiert ist.
6. Fügen Sie der Datei appsettings.json das neue zu verwendende Zertifikat hinzu (siehe **Zertifikat verwenden**).
7. Starten Sie den Dienst neu.
8. Rufen Sie die Swagger-Webseite über den neuen Domänennamen auf, um sicherzustellen, dass das Zertifikat verwendet wird (siehe **HTTPS REST API**).

Netzwerk-Konfiguration

Um ein TLS-Zertifikat für eine Domäne zu erstellen und zu verwalten, müssen Sie Ihr Netzwerk richtig konfigurieren. Der IPv4-Listener des Let's Encrypt-Moduls reagiert auf Port 13020. Der Dienst führt eine HTTP-01-Abfrage durch, um die

Zertifikatsanforderung zu validieren.

Die HTTP-01-Abfrage kann nur auf Port 80 durchgeführt werden. Sie müssen daher auf Ihrem Internet-Router/Firewall eine Portweiterleitung von Port 80 auf den internen Port 13020 des Web API Servers konfigurieren.

Der Server muss Port 13020 für diese Kommunikation öffnen.

Let's Encrypt Konfiguration

Um das Let's Encrypt-Modul zu aktivieren, müssen Sie die Datei appsettings.json konfigurieren.

Konfigurieren Sie den Abschnitt `LetsEncrypt` und setzen Sie den Parameter `active` auf `true`. Ein Beispiel kann wie folgt aussehen:

```
"LetsEncrypt": {
    "active": true,
    "staging": false,
    "user": "anybody.surname@company.com",
    "domain": "sample.dns.net",
    "chronSchedule": "0 0 * * * ?",
    "renewbeforeexpireddays": 30
  },
```

Sie können folgende Parameter im Abschnitt `LetsEncrypt` in der Datei appsettings.json konfigurieren:

Einstellung	Beschreibung
activ	Um das Let's Encrypt-Modul (ACME-Modul) zu aktivieren, setzen Sie diesen Parameter auf `true`. Um das generierte Zertifikat zu verwenden, ändern Sie den Subject-Parameter von `HttpsInlineCertStore` in den Domänennamen, nachdem das Zertifikat erfolgreich generiert wurde.
staging	Let's Encrypt bietet eine Testumgebung für Testzwecke. Wenn Sie diesen Parameter auf `true` setzen, kommuniziert der Dienst mit dieser Testumgebung.

Einstellung	Beschreibung
	Verwenden Sie diesen Parameter nicht zur Erstellung von Zertifikaten. Wenn Sie die Testumgebung verwenden, wird ein ungültiges Zertifikat erstellt, aber nicht installiert. Um ein gültiges Zertifikat anzufordern, setzen Sie den Parameter auf `false`. **Beachten Sie, dass die Anzahl der Zertifikate für einen Domainnamen durch Let's Encrypt begrenzt ist.**
user	Die E-Mail-Adresse des Benutzers, der ein Zertifikat anfordert. Das Modul erstellt entweder ein neues Konto oder verwendet automatisch das bereits bestehende Konto. Die E-Mail-Adresse wird von dem Dienst Let's Encrypt verwendet. Weitere Informationen finden Sie unter: **https://letsencrypt.org/docs/expiration-emails/**
domain	Das vom Dienst verwaltete Zertifikat wird für die angegebene Domäne ausgestellt. Es ist auch das Subject des Zertifikats, wenn es installiert wird, und wird verwendet, um die Zertifikate in der Aktualisierungsroutine des Moduls zu identifizieren.
chronSchedule	Der Dienst überprüft regelmäßig die Gültigkeit des Zertifikats. Der Parameter `chronSchedule` legt die Zeitspanne fest, in der die Überprüfungen durchgeführt werden. Standardmäßig wird eine Überprüfung stündlich durchgeführt. Beim Start des Dienstes wird das Zertifikat einmal überprüft und gegebenenfalls erstellt oder aktualisiert.
renewbeforeexpireddays	Der Parameter `renewbeforeexpireddays` bestimmt, wie viele Tage vor Ablauf das Zertifikat automatisch erneuert wird.

Verwenden Sie die Testumgebung von Let's Encrypt, um sicherzustellen, dass alle anderen Einstellungen und die Netzwerkkonfiguration korrekt sind. Anschließend können Sie die Testumgebung sicher deaktivieren. Weitere Informationen zur Verwendung von Let's Encrypt finden Sie in den Web API Protokolldateien (siehe Service-Protokolldateien).

Zertifikat verwenden

Um ein Zertifikat auszuwählen, konfigurieren Sie den Abschnitt `HttpsInlineCertStore` in der Datei appsettings.json. Der Dienst lädt das Zertifikat einmal beim Start. Um das Zertifikat zu verwenden, setzen Sie den Parameter `Subject` auf das Subject des Zertifikats. Standardmäßig wird das selbstsignierte Zertifikat geladen.

Erstellen Sie zunächst ein Zertifikat mit Let's Encrypt und passen Sie dann den `Subject` für `HttpsInlineCertStore` an. Solange kein Zertifikat gefunden wird, wird der Web API Dienst nicht gestartet.

Das Let's Encrypt-Modul erstellt ein Zertifikat für den Domänennamen Ihres Systems. Diese Domäne ist auch das Subject des generierten Zertifikats.

Stellen Sie sicher, dass der Parameter `Url` nicht für eine bestimmte Domäne konfiguriert ist. Verwenden Sie `https://*:13333` oder `https://0.0.0.0:13333` (Port 13333 kann in einen beliebigen anderen Port geändert werden), um eine beliebige Domäne des Servers abzuhören. Der Dienst ist lokal über localhost (127.0.0.1) oder extern über den konfigurierten Domainnamen erreichbar.

```
"HttpsInlineCertStore": {
   "Url": "https://*:13333",
   "Certificate": {
     "Subject": "sample.dns.net",
     "Store": "My",
     "Location": "LocalMachine"
   }
 },
```

Weitere Informationen

Lokale Portweiterleitung

Wenn Sie eine lokale Portweiterleitung auf dem System benötigen, auf dem die Web API installiert ist, verwenden Sie den folgenden cmd-Befehl:

```
netsh interface portproxy add v4tov4 listenport=80 lis-
tenaddress=0.0.0.0 connectport=13020 connectaddress=127.0.0.1
```

DynDNS

Sie können einen DynDNS-Dienst nutzen, um einen Domänennamen für Ihre Einwahlverbindungen zu erhalten. Die meisten Router unterstützen mehrere DnyDNS-Dienste. DuckDNS.org ist beispielsweise ein kostenloser und benutzerfreundlicher Dienst, den Sie dafür nutzen können.

Fritz Box Benutzer

Wenn Sie die MyFritz-Funktion Ihrer Fritz!Box aktivieren, hat die Fritz!Box einen integrierten Domainnamen. Damit erhalten Sie einen Domänennamen für Ihre Einwahlverbindung, z. B.: `1234abcd.myfritz.net`.

Um diesen Domainnamen verwenden zu können, müssen Sie IPv6 im Internetanschluss deaktivieren. Andernfalls funktioniert die Portweiterleitung von Port 80 auf IPv4 nicht und leitet Sie immer auf die Anmeldeseite der Fritz Box um.

Warum sind HTTPS-Zertifikate wichtig?

Die Erstellung eines SSL-Zertifikats für eine HTTPS-Verbindung trägt dazu bei, die Sicherheit und den Datenschutz Ihrer Inhalte zu gewährleisten und die Sicherheitsstandards für Cyber-Security zu erfüllen.

Vollständige Verschlüsselung der übertragenen Daten

Eine HTTPS-Verbindung mit SSL-Zertifikat bietet eine zusätzliche Sicherheitsstufe für Ihr G-Core System sowie für den Operator. Durch das SSL-Zertifikat werden die über das Internet übertragenen Daten vollständig verschlüsselt, um die sensiblen Informationen vor dem Abfangen und der Manipulation durch Dritte zu schützen.

Sichere Verbindung in Webbrowsern

In weitgehend allen Webbrowsern werden HTTP-Verbindungen als unsicher ausgewiesen. Der Webbrowser Firefox bietet sogar die erweiterte Sicherheitsfunktion "Nur-HTTPS-Modus". Tendenziell werden HTTP-Verbindungen künftig nicht mehr zugelassen oder der Zugriff stark eingeschränkt.

Eigenverantwortung und Kontrolle über Ihre Inhalte

Indem Sie Ihre eigenen SSL-Zertifikate generieren, behalten Sie die Eigenverantwortung und Kontrolle über Ihre Inhalte. Dies gewährleistet, dass Ihre Inhalte sicher und entsprechend Ihren spezifischen Anforderungen übertragen werden.

Einstellungen

Die wichtigsten Einstellungen für den G-Core Web API Service finden Sie in der Datei `appsettings.json` im Web API Installationsordner (`C:\Program Files\Geutebrueck\GCore Web API`).

Verbindung

Name	Typ	Beschreibung	Beispiel
Connection:Address	string	IP-Adresse des G-Core Servers.	127.0.0.1
Connection:username	string	G-Core Benutzername für die Authentifizierung beim Web API Service. Wenn leer, wird die automatische Anmeldung verwendet.	sysadmin
Connection:password	string	G-Core Passwort für die Authentifizierung beim Web API Service. Wenn leer, wird die automatische Anmeldung verwendet.	masterkey

Automatische Anmeldung

Wenn der Benutzername und das Passwort leer sind, verwendet der G-Core Web API Service die automatische Anmeldefunktion, um sich mit dem G-Core Server zu verbinden. Die automatische Anmeldung funktioniert nur wenn G-Core auf demselben Server wie die Web API installiert ist.

Zugriff auf die Web API

Die Web API ist in zwei Hauptteile unterteilt:

- Die **HTTPS REST API** wird für Anfragen wie Authentifizierung oder Ressourcen verwendet.
- Die **WebSocket API** wird für verbundene Anfragen wie Videostreaming oder PLC-Daten verwendet.

HTTPS REST API

Die HTTPS REST API ist mit Swagger dokumentiert. Sie können die Dokumentation über `https://<server-ip>:13333/swagger/index.html` (zum Beispiel: `https://127.0.0.1:13333/swagger/index.html`) in jedem modernen Browser (Chrome, Safari, Firefox und Edge) aufrufen.

Die G-Core Web API wird als Service installiert und enthält eine Swagger-API, über die Entwickler die Beschreibung der vollständigen Schnittstelle abrufen können (https://localhost:13333/swagger/index.html).

Sie können einen HTTP-Webclient erstellen und auf diese GET- und POST-Befehle zugreifen, oder Sie können eine c#-Anwendung erstellen und die eingeschlossene HTTP-Funktion in einer Klasse verwenden.

WebSocket API

Die WebSocket API verwendet das WebSocket-Protokoll auf der Grundlage einer TCP-Verbindung.

Die entsprechende WebSocket-API-Dokumentation kann über folgende URLs aufgerufen werden:

- **Streaming:** `https://<server-ip>:13333/asyncapi/media/ui/index.html`
- **PLC:** `https://<server-ip>:13333/asyncapi/plc/ui/index.html`

Auf diesen Seiten finden Sie die notwendige Dokumentation für die Verwendung der WebSocket-API-Endpunkte.

Authentifizierung

Verwenden Sie den Authentifizierungsendpunkt **/api/1/Login**, um sich bei der G-Core Web API zu authentifizieren. Der G-Core Benutzername und das Passwort werden zur Authentifizierung verwendet.

Authentifizierung in Swagger

Sie können die Authentifizierung direkt in der Swagger-Benutzeroberfläche durchführen. Das Ergebnis ist ein Tupel, das Sie anschließend in allen anderen zu authentifizierenden Anfragen verwenden sollten. Das Tupel besteht aus einem AccessToken und einem RefreshToken.

Token	Gültigkeit	Verwendung
AccessToken	Gültig für einen Zeitraum von 15 Minuten.	Bei HTTP-Anfragen kann das Token im Authorization-Header verwendet werden. Zum Beispiel: "Autorisierung: Bearer <AccessToken>"
RefreshToken	Gültig für 7 Tage. **Jedes RefreshToken ist nur einmal gültig.**	Das Token kann verwendet werden, um neue Zugangstoken über den Endpunkt **/api/1/ RefreshLogin** zu erlangen, der ebenfalls ein Tupel aus AccessToken und RefreshToken zurückgibt.

Der Authentifizierungsendpunkt /api/1/Login ist in den Standardeinstellungen auf eine bestimmte Anzahl von Anfragen begrenzt. Pro Client-IP sind 100 Anfragen pro 10 Minuten zulässig. Diese Einstellung kann in der appsettings.json Datei im Abschnitt IpRateLimiting konfiguriert werden.

Geben Sie in Swagger das Token ein, um die Authentifizierung durchzuführen:

1. Klicken Sie in Swagger in der oberen rechten Ecke auf **Autorisieren**.
2. Geben Sie das Wort **Bearer** gefolgt von einem Leerzeichen und dem Token ein, das Sie vom Anmeldeendpunkt erhalten haben (Bearer <AccessToken>).
3. Dann können Sie jeden anderen Endpunkt in Swagger ausprobieren.

→ Diese Authentifizierung ist auch für die WebSocket-API erforderlich.

Authentifizierung der WebSocket-API

Es gibt zwei unterstützte Methoden zur Authentifizierung der WebSocket-API:

- Sie können das Token in einem Autorisierungs-Cookie festlegen.

```
ClientWebSocket _webSocket = new ClientWebSocket();
var token = "...";
_webSocket.Options.Cookies = new System.Net.CookieContainer();
_webSocket.Options.Cookies.Add(new Uri(String.Format("ws://
{0}/", host)), new System.Net.Cookie("Authorization", token));
_webSocket.ConnectAsync(
        new Uri(String.Format("ws://{0}/a-
pi/1/stream/video?MediaChannelIdentifier={1}", host, mediaChan-
nel)), cancellationToken).Wait();
```

- Sie können das Token wie im Protokoll-Header einstellen.

```
this.socket = new WebSocket(url, accessToken);
```

Arbeiten mit der Web API

JavaScript WebSocket Beispiel

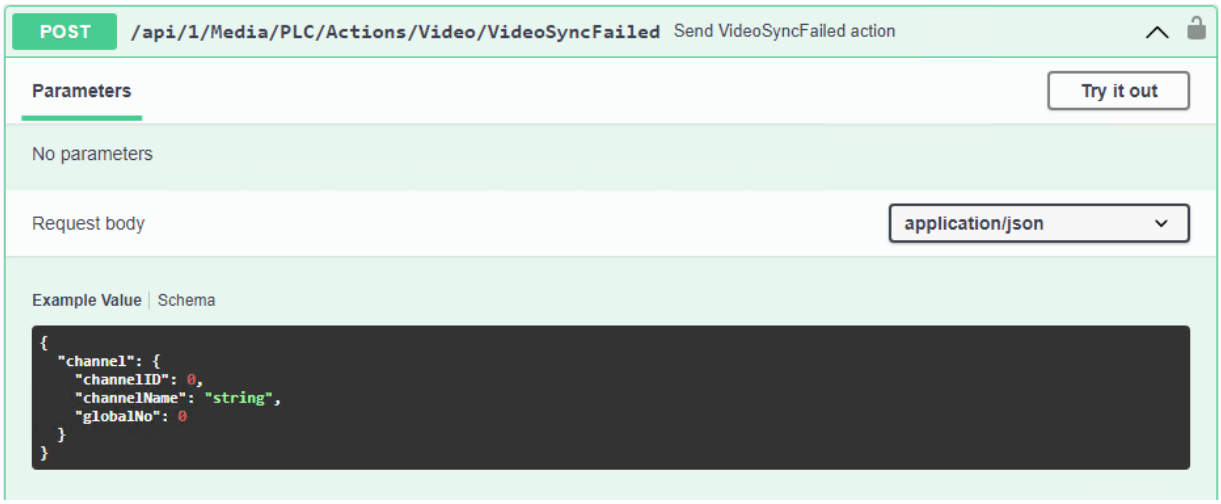
Dies öffnet einen neuen Stream mit der Mediakanal ID-Nummer 1.

```
const websocketAddress =
    "ws://<server ip>:13332/a-
pi/1/stream/video?MediaChannelIdentifier=1";
let websocket = new Websocket(websocketAddress);
```

Parameter "Channel"

Über die Web API können Sie Actions in G-Core auslösen.

Die Actions können in der Swagger-Benutzeroberfläche konfiguriert werden. Viele Actions sind einem Medienkanal zugeordnet. Diese Zuordnung erfolgt über den Parameter `"channel"`:



Sie müssen einen der folgenden Werte für den `"channel"` Parameter angeben:

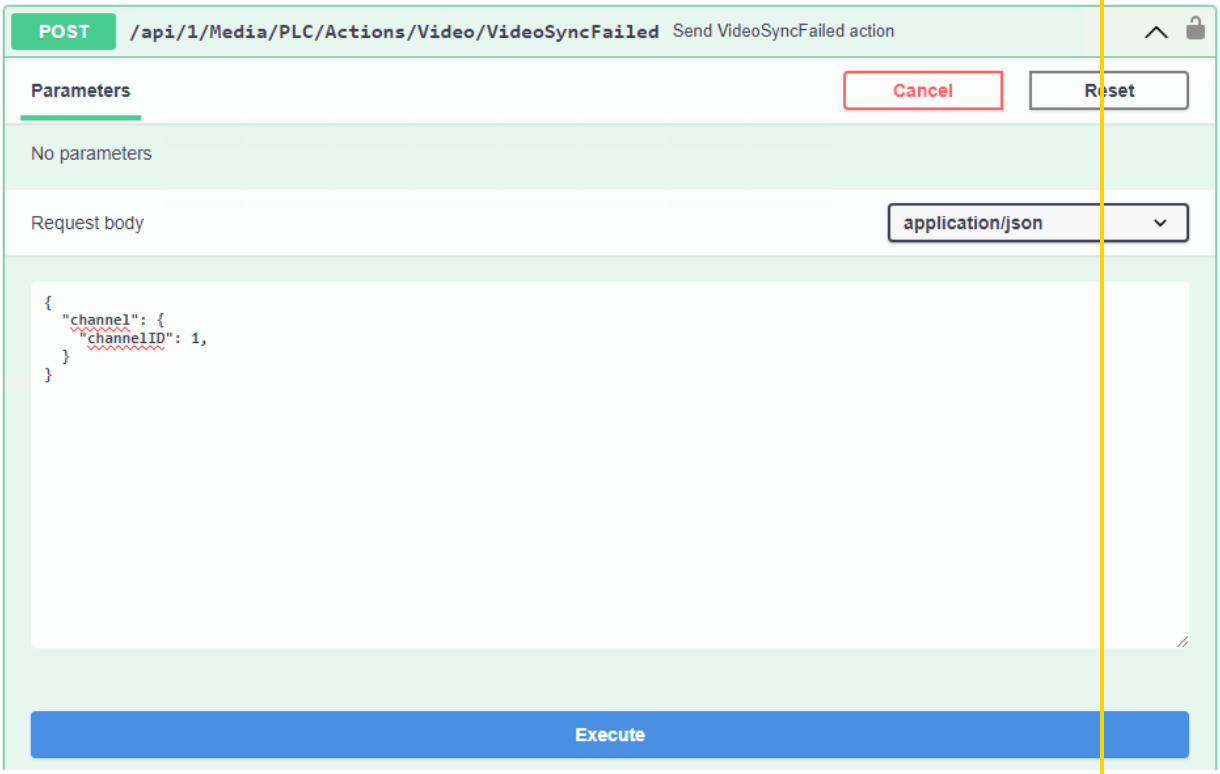
Wert	Beschreibung
"channelID"	Die lokale Nummer des Medienkanals.

Wert	Beschreibung
"channelName"	Der Name des Kanals. Dieser muss genau übereinstimmen, um eine Zuordnung zu gewährleisten.
"globalNo"	Die globale Nummer des Medienkanals.

Geben Sie nur einen Wert für den Parameter an. Wenn Sie mehr als einen Wert angeben, kann G-Core den zugewiesenen Medienkanal nicht finden und die Action wird nicht ausgelöst.

Beispiel

In diesem Beispiel wird der Wert "`channelID`" angegeben. Die anderen Werte müssen entfernt werden, da sie nicht angegeben werden dürfen.



G-Core Action Referenz

In diesem PDF-Dokument finden Sie eine vollständige Definition aller G-Core Actions und Parameter: **G-Core Actions Reference.**

RTSP

RTSP-Streaming verwenden

Um RTSP-Streaming zu verwenden, fügen Sie den folgenden Abschnitt in die Datei appsettings.json (`C:\Program Files\Geutebrueck\GCore Web API\appsettings.json`) ein.

Mit diesen Parametern können Sie das RTSP-Streaming (siehe **RTSP-Server**) und das Streaming von Aufzeichnungslücken (siehe **Streaming**) konfigurieren.

```
"Streaming": {
    "DBPlaybackMaximumGapMs": 5000,
    "DBPlaybackGapRecoverMs": 40
  },
  "RTSPServer": {
    "EnableRTSP": true,
    "ListenV6": "[::]",
    "ListenV4": "0.0.0.0",
    "ListenPort": 554
    //,"LogStreamFilePath": "c:\\temp"
  },
```

RTSP-Server

Um RTSP-Streaming zu aktivieren, muss der Parameter **EnableRTSP** auf **true** gesetzt sein.

Verschlüsselung des RTSP-Streams

Sie können den RTSP-Stream mit einer VPN-Verbindung zwischen dem Client und der Web API verschlüsseln.

Wenn Sie nur den RTSP-Stream verwenden, ist ein SSL-Tunnel auch ausreichend.

Authentifizierung

Der Client sendet den Benutzernamen und das Passwort an den G-Core Server, von dem der Client den Stream erwartet. Diese Anmeldedaten werden dann verwendet, um den Benutzer für den Stream zu authentifizieren. Wenn die Authentifizierung fehlschlägt, wird eine RTSP 401-Antwort gesendet.

Es wird nur die Basisauthentifizierung unterstützt.

WICHTIG! Diese Art der Benutzerauthentifizierung ist nicht sicher und wird nicht empfohlen.

Aufgezeichneter und Live-Kanal

Die Web API unterscheidet zwischen der Wiedergabe von aufgezeichneten und Live-Kanälen.

Aufgezeichneter Kanal

Wenn ein bestimmter Zeitrahmen abgerufen wird, prüft die Web API, ob der angeforderte Zeitrahmen existiert, und spielt ihn mit der normalen 1,0x-Geschwindigkeit ab.

Beispiel

```
rtsp://localhost?MediaChannelIdentifier=1&Start=2022-01-31T13:05:04.447&End=2022-01-31T14:05:04.447
```

`Start=` Startzeit im Format `year-month-day T hours : minutes : seconds`.

`End=` Endzeit im Format `year-month-day T hours : minutes : seconds`.

Live-Kanal

Wenn weder eine Start- noch eine Endzeit angefordert wird, spielt die Web API den angeforderten Medienkanal so schnell wie möglich und äquivalent zur Live-Ansicht ab.

Beispiel

```
rtsp://localhost?MediaChannelIdentifier=1
```

Streaming

Wenn Lücken in der Datenbank vorhanden sind, werden diese bei der Wiedergabe übersprungen und das nächste verfügbare Bild wird angezeigt. Sie können die Wiedergabe von Aufzeichnungslücken im Abschnitt `Streaming`

konfigurieren.

Parameter	Beschreibung
DBPlaybackMaximumGapMs	Definiert in Millisekunden, wie groß eine Lücke ist, die nicht als Lücke angesehen wird.
DBPlaybackGapRecoverMs	Definiert in Millisekunden, nach welcher Zeit das nächste verfügbare Bild abgespielt wird.

Leistung

Um die Grenzen des G-Core Web SDK und der Web API sowie einige Funktionalitäten zu testen, wurden einige Leistungsmessungen durchgeführt.

Spezifikationen

Die folgenden Spezifikationen haben wir für die Leistungsmessung verwendet:

- **Betriebssystem:** Windows 10 x64
- **CPU:** Intel i7-7700
- **RAM:** 8 GB

Testergebnisse

Szenario: Full HD 12,5 fps / im Freien ganz

Um die Testergebnisse vergleichbar zu machen, betrug die Dauer der einzelnen Messungen jeweils ~10 Minuten:

Anzahl der Viewer	CPU-Nutzung	Speicherverbrauch	Testergebnis
4	~20%	~45%	Bestanden (stabil)
8	~20%	~60%	Bestanden (stabil)
10	~20%	~80%	Bestanden (stabil)
14	~40%	~70%	Bestanden (stabil)
18	~50%	~80%	Fehlgeschlagen (instabil)

Framerate
Wir haben festgestellt, dass einige RTSP-Clients Probleme mit sehr niedrigen Bildraten haben. Wir empfehlen daher eine Bildrate von mindestens 5 FPS.

Service-Protokolldateien

Die Web API protokolliert die wichtigsten Meldungen im Windows-Ereignisprotokoll.

Eine ausführlichere Protokolldatei finden Sie hier: `%PROGRAMDATA%\Geutebrueck\GCoreWebApi\GCoreWebApiLog.log`.

Alle Meldungen des Dienstes werden in dieser Datei protokolliert.

Technische Änderungen vorbehalten.

GEUTEBRÜCK GmbH
Im Nassen 7-9 | D-53578 Windhagen
Tel. +49 (0)2645 137-0 | Fax-999
info@geutebrueck.com

www.geutebrueck.com