

第四回 : EagleEye APIのご紹介



いつでも、どこでも、簡単、安心
クラウドカメラ

EAGLE EYE
NETWORKS, INC.

サーバー不要 / ソフトウェア導入不要 / 容易な設置

株式会社イグアズ
テクニカル&クラウド事業部

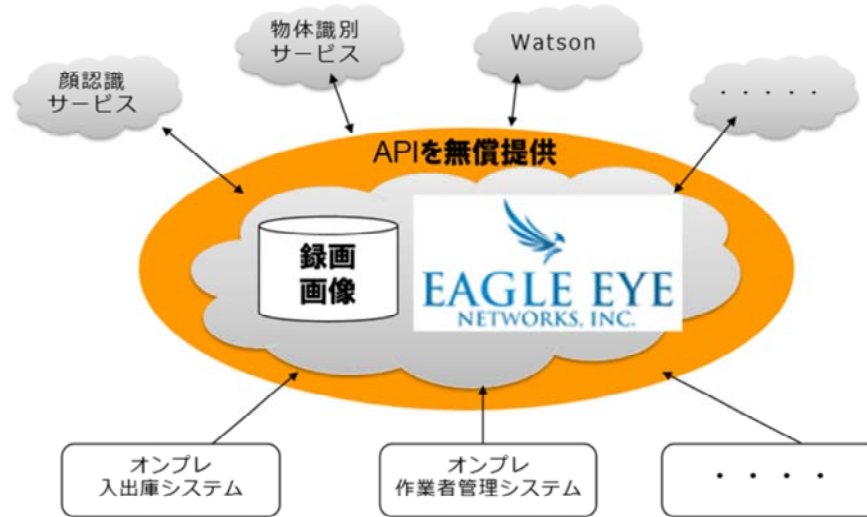
iGUAZU

Copyright 2016 iGUAZU Corporation

いつでも、どこでも、簡単、安心、クラウドカメライーグルアイのご紹介。第四回、EagleEye APIのご紹介

EagleEyeの強みはAPIで「つながる！」

外付けの録画・再生装置としてEagleEyeを活用 アプリケーションに「目」と「耳」を追加



Copyright 2016 iGUAZU Corporation

1

EagleEye長所は、これまでの録画システムと比較して、録画データの損失が低いことや、いつでも、どこからでも利用が可能な、クラウドによる利点だけではありません。

EagleEyeの最大の強みはAPIで「つながる！」事です。

この「つながるAPI」を用いて、例えばEagleEyeを外付けの録画再生装置のように活用し、業務アプリケーションに目と耳の追加が可能です。

業務アプリケーションには、あの時、あの瞬間、何が行われていたのか、動画や音声で確認ができると大変助かる場面が多くあります。決定的瞬間を社内、社外の関係者と直ぐに共有し、共通認識を得ることで信用・信頼の向上とビジネススピードの加速を同時に支援します。

EagleEye APIの特徴

①簡単

-APIはREST原理に基づいているので、作成やアプリケーションのテストが非常に簡単

②安全

-APIの実行にはユーザー認証が必要で、認証から実行、データの取得まで全てHTTPSで実行されるため、通信の全てが暗号化されます。

③便利

-APIは録画の管理や任意のカメラのライブビュー、ビデオ再生を時と場所を選ばずに行えるよう、アプリケーションに組み込みます。

EagleEye APIの特徴は、①簡単、②安全、③便利である事です。

APIはREST原理に基づいているので、SDKと比較して、連携する側、される側に何かを導入、設定をする必要少なく、<http://~> といったURLを呼び出すため、簡単に利用する事が可能です。

APIの実行にはユーザー認証が必要で、認証から実行、データの取得まで、全て通信はHTTPSで暗号化されているため安全です。

APIは録画の管理や任意のカメラのライブビュー、ビデオ再生を時と場所を選ばずに行えるよう、アプリケーションに組み込む事が可能です。

EagleEye APIで出来る事:APIドキュメント

APIドキュメント 1部日本語

<http://een-api-doc-jp.mybluemix.net/>

APIドキュメント 最新版(英語)

<https://apidocs.eagleeyenetworks.com/apidocs/>

認証、認可

ユーザー

カメラ

ブリッジ

画像と動画

ポーリング

レイアウト

アカウント

アクション

アノテーション(注釈)

フィードバック

メトリック

PNGスパン

録画

検索

エラー

APIで実現可能な事はご覧のURLリンクを参照頂くか、または「イーグルアイ API」で検索をお願いいたします。

前提環境

Windowsの場合Gitを導入しBashを利用

-<https://git-scm.com/download/win>

JSON成形のためjqを利用

-<https://stedolan.github.io/jq/>

インターネットに直接接続

-プロキシ経由ではありません

APIドキュメントでの解説例はcURLを用いていますので、Bashが使えると便利です。

Windowsの場合はGitを導入するとBashが使えるようになります。

また今回はJSONを成形するためjqを用いております。

インターネットに直接接続している環境が前提となっています。プロキシを経由する場合cURLコマンドにオプションが必要となります。

APIの使い方(1):認証

API Reference

Search

- はじめに
- 認証
- 概要
- ステップ 1: 認証
- Step 2: 認可
- ユーザー
- カメラ
- アプリ
- 画像と動画
- ホーリング
- レイアウト
- AAA
- 利用規約
- アカウント
- アクション
- アクション(注釈)
- フィードバック

ステップ 1: 認証

ログインは2ステップのプロセスがあります: 認証と認証時に送られたトークンを使用した認可です。

HTTPリクエスト

POST <https://login.eagleeyenetworks.com/g/aaa/authenticate>

パラメータ	データ形式
username	文字列
password	文字列

エラー ステータスコード

HTTP ステータスコード	データ形式
400	いくつかの引数が不足しているか不正です
401	与えられた認証情報が不正です
402	アカウントは休止中です
460	アカウントは非アクティブです
461	アカウントは保留中です

cURL

要求

```
curl -v --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate --data-urlencode "username=[USERNAME]" --data-urlencode "password=[PASSWORD]"
```

JSON応答

```
{
  "token":
  "03k8hE3jQgJaxC8bL69/5cH+Z7eMdPe8c+UpEZHX0s7PTFH450r9M3wxLkP66jcPuCw81XVTKH
  GA1zgx/q44H8v3XmcJ4/XzN2f6Hv+mZVIy8LorX8N5a6fNVRknaM86nOHfbLvOP6TPcm8P1dD18
  ymr6eAd1Qhtq9S5evKh24Ma2gVfbM4Dyhyku+eTN+11XN80egJdZ8jhaYCZ1FVKkdnr1sr9D6J5r
  /tE7byCLVjPcwcVabA+8btD061pystTNYZyDvr-30Qn705V6xfq21r1CB/zDu7a2Eh111QwZ22
  GQ1Qm5j8t-j9UR/p7a1nHVhEe/b5FYUCvz1epcAa=="
}
```

APIドキュメントの「認証」の「ステップ 1: 認証」を御覧ください

APIの使い方(1): 認証

認証

```
curl -v --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate --data-urlencode "username= [USERNAME]" --data-urlencode "password= [PASSWORD]"
```

JSON応答

```
{  
  "token":  
    "O3k0hNH3jQgjaxC0bLG9/5cM+Z7eWdPe0c+UpEZXOs7PTFH45Dr9M3wxLkP6GjcPuC  
w8IXVTkHGA1zgx/q44HBv3Xmcj4/XzN2f6Hv+mZVly8LorX8N5a6fNVRknWWW86nCHfbL  
vOP6TPcmBP1dD1OynnGeAdlQHTqMN5mvKH24WwZgVFbM4DyhyWu+eTN+t1XNROegJdZ  
RjhaYCZ1FVKkdnrlsrMD6JSr/tE7byCLVjPcwzVabA+x0tDbGlpystTNYPZyDvr3DQM7OSV6k  
fqg2iriC8/zDu7a2Ehl1IQWuZZ2GQIQm5jBtj9UR/p7ainHVhEc/bSFYUCvziepcAa=="  
}
```

EagleEyeをAPIでアクセスするには認証と認可が必要です。

APIの使い方(1):認証

認証

```
curl -v --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate --data-urlencode "username= [USERNAME]" --data-urlencode "password= [PASSWORD]"
```

JSON応答

```
{  
  "token":  
    "O3k0hNH3jQgjaxC0bLG9/5cM+Z7eWdPe0c+UpEZNXOs7PTFH45Dr9M3wxLkP6GjcPuC  
w8IXVTkHGA1zgx/q44HBv3Xmcj4/XzN2f6Hv+mZVly8LorX8N5a6fNVRknWWW86nCHfbL  
vOP6TPcmBP1dD1OynnGeAdlQHTqMN5mvKH24WwZgVfBm4DyhyWu+eTN+t1XNROegJdZ  
RjhaYCZ1FVKkdnrlsrMD6JSr/tE7byCLVjPcwzVabA+xOtDbGipystTNYPZyDvr3DQM7OSV6k  
fqg2irIC8/zDu7a2Ehl1IQWuZZ2GQIQm5jBtj9UR/p7ainHVhEc/bSFYUCvziepcAa=="  
}
```

クライアントは初めに使い捨ての認証トークンを取得するために

APIの使い方(1): 認証

認証

```
curl -v --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate --data-urlencode "username= [USERNAME]" --data-urlencode "password= [PASSWORD]"
```

JSON応答

```
{  
  "token":  
    "O3k0hNH3jQgjaxC0bLG9/5cM+Z7eWdPe0c+UpEZNXOs7PTFH45Dr9M3wxLkP6GjcPuC  
w8IXVTkHGA1zgx/q44HBv3Xmcj4/XzN2f6Hv+mZVly8LorX8N5a6fNVRknWWW86nCHfbL  
vOP6TPcmBP1dD1OynnGeAdlQHTqMN5mvKH24WwZgVfBm4DyhyWu+eTN+t1XNROegJdZ  
RjhaYCZ1FVKkdnrlsrMD6JSr/tE7byCLVjPcwzVabA+xOtDbGlpystTNYPZyDvr3DQM7OSV6k  
fqg2iriC8/zDu7a2Ehl1IQWuZZ2GQIQm5jBtj9UR/p7ainHVhEc/bSFYUCvziepcAa=="  
}
```

ユーザーIDとパスワードを指定し、指定のURLへ要求をします。

APIの使い方(1):認証

認証

```
curl -v --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate --data-urlencode "username= [USERNAME]" --data-urlencode "password= [PASSWORD]"
```

JSON応答

```
{  
  "token":  
    "O3k0hNH3jQgjaxC0bLG9/5cM+Z7eWdPe0c+UpEZXOs7PTFH45Dr9M3wxLkP6GjcPuC  
w8IXVTkHGA1zgx/q44HBv3Xmcj4/XzN2f6Hv+mZVly8LorX8N5a6fNVRknWWW86nCHfbL  
vOP6TPcmBP1dD1OynnGeAdIQHTqMN5mvKH24WwZgVFbM4DyhyWu+eTN+t1XNROegJdZ  
RjhaYCZ1FVKkdnrlsrMD6JSr/tE7byCLVjPcwzVabA+xOtDbGipystTNYPZyDvr3DQM7OSV6k  
fqg2iriC8/zDu7a2Ehl1IQWuZZ2GQIQm5jBtj9UR/p7ainHVhEc/bSFYUCvziepcAa=="  
}
```

JSON応答に含まれる、この使い捨てトークンを用いて、次に認可をします。

この使い捨てトークンは発行されて30秒間だけ有効ですので速やかに認可を行う必要があります。

APIの使い方(2):認可

Step 2: 認可

認可はログイン プロセスの2ステップ目で、最初のステップ(認証)で作成されたトークンを使用します。この応答はユーザーオブジェクトの認可を行い、'auth_key'クッキーに設定します。この値に実行するAPIコールでは、可能であればクッキー、またはクッキー内の値を 'A' パラメータとしての、いずれかの方法で送信します。

APIコールを行うホストURLはオリジナルの "https://login.eagleeyenetworks.com" に対して行いますが、APIは認可後に返される **ブランド サブドメイン** に対して実行しなければなりません。ブランドホストURLは "https://[アクティブなブランド サブドメイン].eagleeyenetworks.com" となり、**アクティブなブランド サブドメイン** フィールドは認可の応答で返されます。

認可後の例が右側に表示されていますが、ホストURLは "https://c001.eagleeyenetworks.com" に変更されています。

それぞれのアカウントは常に同じ **ブランド サブドメイン** が使用され、同様に同一セッション中も変更されません。サブドメインのキャッシュは、認可後の active_brand_subdomain に対してクライアント ソフトウェアの確認をより長期にわたって安全に保ちます。ブランド サブドメインを使用することは、速度と信頼性上重要です。

HTTP要求

```
POST https://login.eagleeyenetworks.com/g/aaa/authorize
```

パラメータ	データ形式
token	文字列

cURL

```
curl -D - --request POST https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode token=[TOKEN]
```

JSON応答

```
{  "is_sms_include_picture": 0,  "last_name": "",  "uid": " ",  "is_export_video": 1,  "layouts": [    "217f0f04-450f-11e4-a983-ca8312ea370c"  ],  "account_map_lines": null,  "postal_code": null,  "is_account_superuser": 1,  "timezone": "US/Pacific",  ...}
```

APIドキュメントの「認証」の「ステップ 2: 認可」を御覧ください

APIの使い方(2):認可

認可

```
curl -D - --request POST https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode token= [TOKEN]
```

JSON応答

```
{  
  "is_sms_include_picture": 0,  
  "last_name": "",  
  "uid": " ",  
  "is_export_video": 1,  
  "layouts": [  
    "217f0fd4-450f-11e4-a983-ca8312ea370c"  ]  
}
```

認可により認証したユーザーに対する操作、参照の範囲など・・俗に言う権限の適用を実施します。

APIの使い方(2):認可

認可

```
curl -D - --request POST https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode token=[TOKEN]
```

JSON応答

```
{  
  "is_sms_include_picture": 0,  
  "last_name": "",  
  "uid": " ",  
  "is_export_video": 1,  
  "layouts": [  
    "217f0fd4-450f-11e4-a983-ca8312ea370c"  ]  
}
```

括弧内の[TOKEN]が前手順の「認証」の結果、JSON応答で得られた使い捨てトークンが入ります。

APIの使い方(3): 認証、認可をする

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

```
token=`curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token`
```

```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

Copyright 2016 IGUAZU Corporation

13

認証と認可を実行してみましょう。

例えば・・・画面例のようなシェルで認証と認可を実行し、auth_keyの取得が可能です。

APIの使い方(3): 認証、認可をする

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

```
token=`curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token`
```

```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

EagleEyeへログインに使用しているメールアドレスとパスワードを指定し認証をします。

APIの使い方(3): 認証、認可をする

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

```
token=$(curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token)
```

```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

認証のJSON応答に対してjqコマンドを使用して、使い捨てtokenの部分だけを取得しています。

APIの使い方(3): 認証、認可をする

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

```
token=`curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token`
```

```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

確認のため取得した使い捨てトークンを画面に表示しています。

APIの使い方(3): 認証、認可をする

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

```
token=`curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token`
```

```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

使い捨てトークンをセットし

APIの使い方(3): 認証、認可をする

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

```
token=`curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token`
```

```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

クッキーをcookie.txtとして保存するようにし、認可リクエストをしています。

APIの使い方(3): 認証、認可をする

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

```
token=`curl --request POST https://login.eagleeyenetworks.com/g/aaa/authenticate  
--data "username=test01@test.co.jp&password=password01" --insecure |  
./jq.exe -r .token`
```

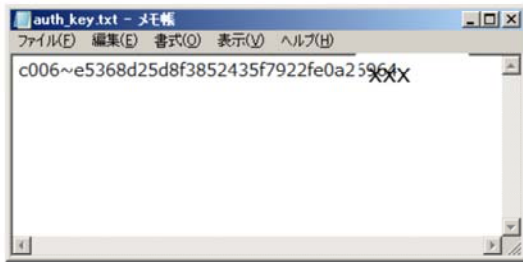
```
echo $token
```

```
curl -D - -c cookie.txt --request POST  
https://login.eagleeyenetworks.com/g/aaa/authorize --data-urlencode  
token=$token
```

```
grep auth_key cookie.txt | awk '{print $7}' > auth_key.txt
```

保存したcookie.txt内のauth_keyをauth_key.txtとして保存しています。

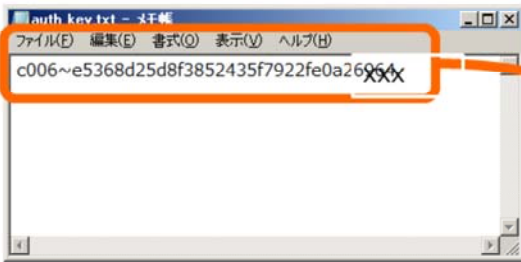
APIの使い方(3): 認証、認可をする



```
auth_key.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
c006~e5368d25d8f3852435f7922fe0a25964XXX
```

実行したフォルダーに保存されているauth_key.txtを参照するとc006~といった文字列が確認できます。

APIの使い方(3): 認証、認可をする



```
curl -v -G  
https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id=[CAMERA_ID];timestamp=[TIMESTAMP];asset_class=[ASSET_CLASS];A=[AUTH_KEY]
```

この値を今後APIを用いる時、Aパラメーターに指定します。

APIの使い方(4): 静止画を取得する

静止画の要求

```
curl -v -G
```

```
https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id= [CAMERA_ID] ;timestamp= [TIMESTAMP] ;asset_class= [ASSET_CLASS] ;A= [AUTH_KEY]
```

それでは「静止画の要求」を例に、APIを使ってみましょう。

「静止画の要求」はAPIドキュメントの画像と動画 (Images and Video) のGet Imageです。

APIの使い方(4): 静止画を取得する

静止画の要求

curl -v -G

```
https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id=[CAMERA_ID];timestamp=[TIMESTAMP];asset_class=[ASSET_CLASS];A=[AUTH_KEY]
```

カメラ設定 // EN-CDUB-001a_00015

カメラ 保存期間 解像度 モーション オーディオ ロケーション メトリクス

名前: EN-CDUB-001a_00015

ログイン: ユーザー名 パスワード

タイムゾーン: Asia/Tokyo

タグ: bullet x een x add a tag

ノート: 2016/6/14 保江追加 API連携テスト用 サーバラック屋根の上に設置

情報: ESN: 10069caf

Get RTSP Info

キャンセル 変更を保存

right 2016 iGUAZU Corporation 23

[CAMERA_ID]はカメラ設定の情報欄のESN値です

APIの使い方(4): 静止画を取得する

静止画の要求

curl -v -G

```
https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id= [CAMERA_ID] ;timestamp=[TIMESTAMP] ;asset_class= [ASSET_CLASS] ;A= [AUTH_KEY]
```

id	文字列	Camera Id	true
timestamp	文字列	Timestamp in EEN format: YYYYMMDDHHMMSS.NNN	true
asset_class	文字列, enum	Asset class of the image enum: all, pre, thumb	true
quality	文字列, enum	Future Feature: Quality of image enum: low, med, high	

見た目の時間から
9時間をひく

[TIMESTAMP]はグリニッジ標準時(GMT時間)で四桁の年、二桁の月、時間、分、秒、さらに3桁を付加し指定します。

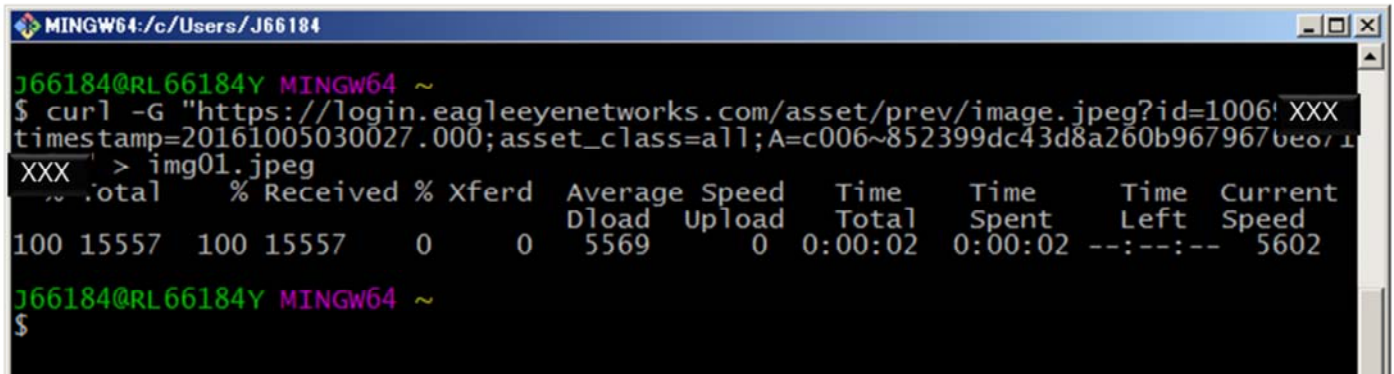
時間指定はグリニッジ標準時(GMT時間)のため、見た目の日本時間から9時間ひく事にご注意して下さい。

APIの使い方(4): 静止画を取得する

静止画の要求例

curl -G

```
"https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id=10069xxx:timestamp=20161005030027.000;asset_class=all;A=c006~852399dc43d8a260b9679676e8712xxx" > img01.jpeg
```



```
MINGW64:/c/Users/J66184
J66184@RL66184Y MINGW64 ~
$ curl -G "https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id=1006:XXX
timestamp=20161005030027.000;asset_class=all;A=c006~852399dc43d8a260b9679676e871
XXX" > img01.jpeg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left     Speed
100 15557  100 15557    0     0   5569      0  0:00:02  0:00:02  --:--:--  5602
J66184@RL66184Y MINGW64 ~
$
```

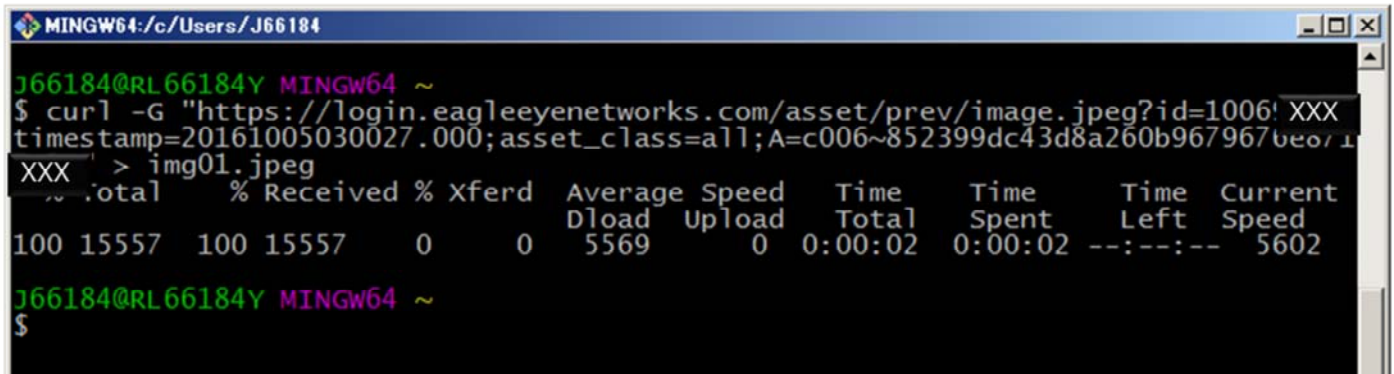
ご覧の「静止画の要求例」では結果をリダイレクトしてimg01.jpgとして保存をしています。

APIの使い方(4): 静止画を取得する

静止画の要求例

curl -G

```
"https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id=10069xxx:timestamp=20161005030027.000;asset_class=all;A=c006~852399dc43d8a260b9679676e8712xxx" > img01.jpeg
```

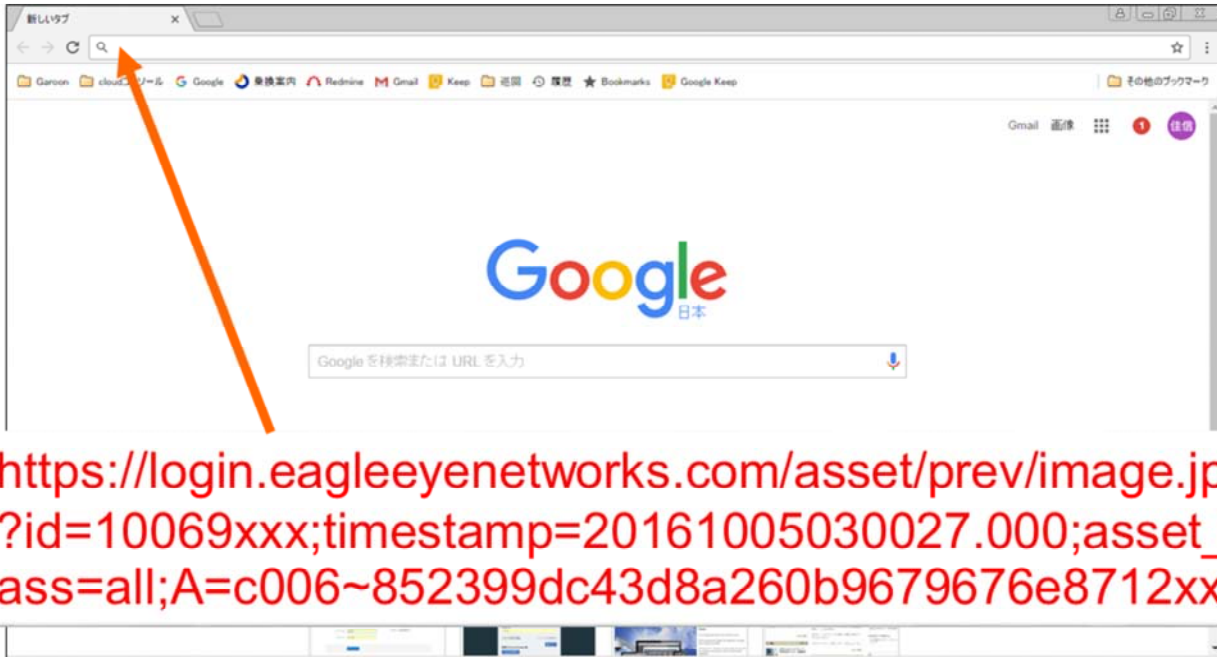


```
MINGW64:/c/Users/J66184
J66184@RL66184Y MINGW64 ~
$ curl -G "https://login.eagleeyenetworks.com/asset/prev/image.jpeg?id=1006:XXX
timestamp=20161005030027.000;asset_class=all;A=c006~852399dc43d8a260b9679676e871
XXX" > img01.jpeg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 15557  100 15557    0     0   5569      0  0:00:02  0:00:02 --:--:--  5602
J66184@RL66184Y MINGW64 ~
$
```

ブラウザを使ったやり方も試してみましょう。

Curl -Gとリダイレクト部分のimg01.jpg以外のhttps://から始まる部分を

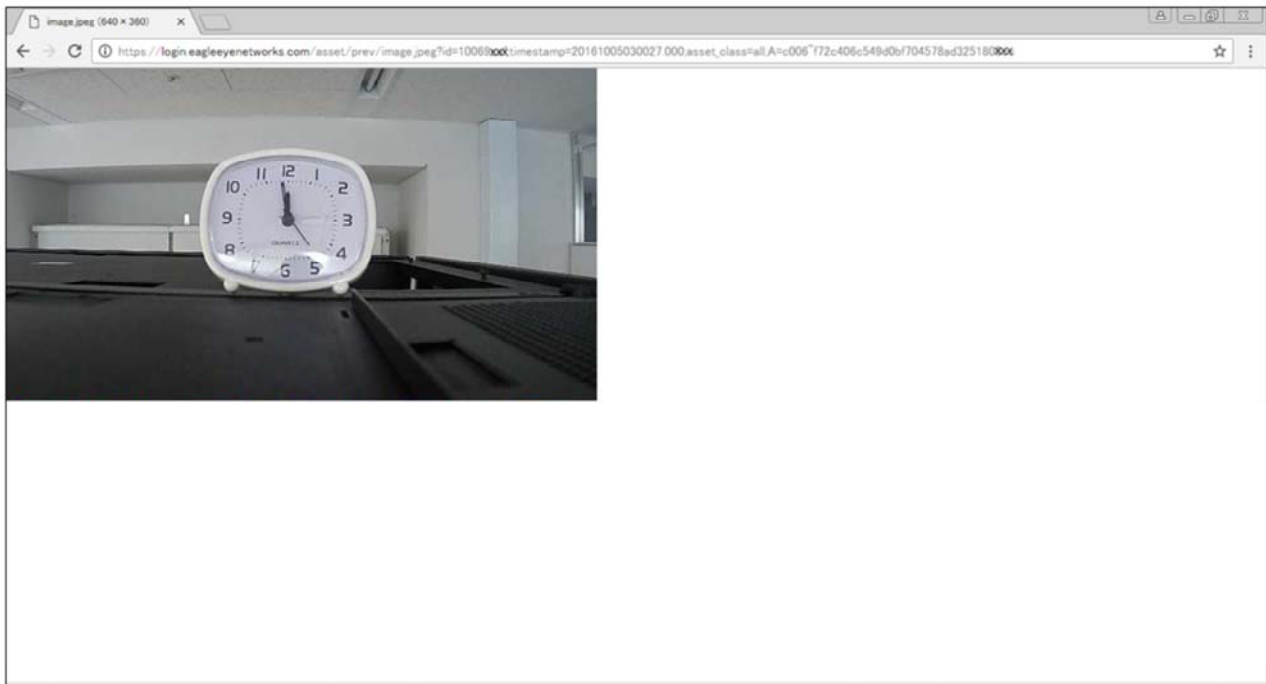
APIの使い方(4): 静止画を取得する



Copyright 2016 IGUAZU Corporation 27

ブラウザのURL欄に入力しEnterキーを押すと

APIの使い方(4): 静止画を取得する



Copyright 2016 iGUAZU Corporation

28

ブラウザに静止画が表示されます。

APIの使い方(5):AUTH_KEYの期限

コントロール 日 セキュリティ カメラ アラート 通知 共有 Responders

ウェブタイムアウト: 24 時間

Inactive Session Timeout: なし

ログイン試行回数:

システム通知に画像を含める:

キャンセル 変更を保存

Copyright 2016 IGUAZU Corporation 29

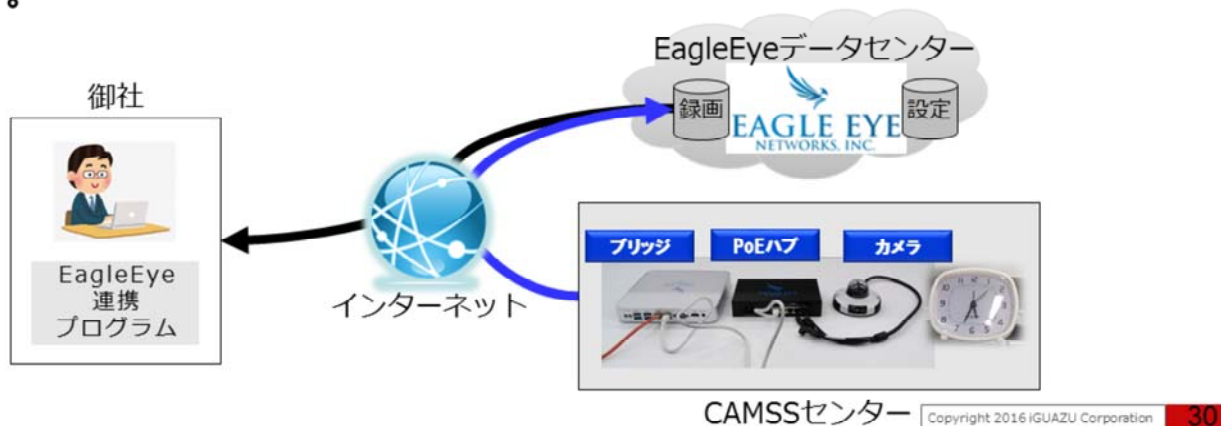
Aパラメータに指定するAUTH_KEYには期限があります。

認証時に指定するユーザーIDのアカウント設定の「セキュリティ」タブの「ウェブタイムアウト」の時間で期限切れとなります。

ウェブタイムアウト値は15分、1時間、4時間、8時間、12時間、24時間、1週間から選択が可能です。

API検証環境のご案内

1. EagleEye機器を御社に設置する事なしに、EagleEye API連携の仕様を体験して頂くプログラムです。
2. 弊社がEagleEye API連携テストに必要な環境(テスト用ユーザーID、パスワード、カメラ...)をご用意させて頂き、御社からインターネットを経由して検証が可能です
3. 連携可能な映像は、弊社カメラが撮影した時計となり、御社が実際に連携する映像とは異なります。



弊社イグアスではAPI連携の検証環境を常設しております。

EagleEye機器を御社に設置する事なしに、EagleEye API連携を体験して頂くプログラムです。

弊社がEagleEye API連携テストに必要な環境をご用意させて頂き、御社からインターネットを経由で検証が可能です。

下記「お問合せ」URLよりご連絡をお待ちしております。

「イグアス イーグルアイ」で検索！

iGUAZU

