Endeavour Health Common Interface Mechanism
Authentication protocol

http://endeavour-cim.cloudapp.net/

Introduction

To make calls to the CIM in your own code, you will need to provide two pieces of authentication data as HTTP headers, otherwise your call will be rejected with an Unauthorised error. The two pieces of information (and their HTTP header names) are:

api_key	API key you received when you signed up for the CIM.
hash	This must be generated each time you make a call.

Hash generation

The hash value is generated using the following formula:

```
hash = ConvertToBase64(ComputeHMACSHA256Hash(secret, dataToHash))
```

The two inputs to the above formula are:

secret	API secret you received when you signed up for the CIM
dataToHash	See below for how to construct this.

Creating the data to hash

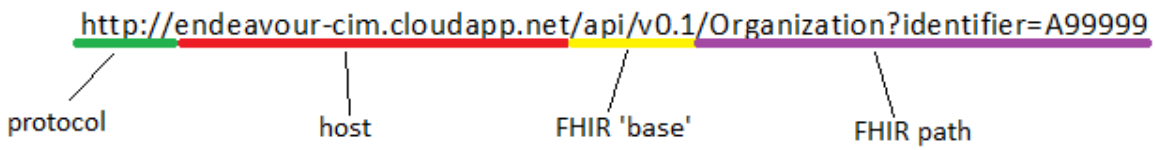
To generate the dataToHash input:

1. **Construct the URL for the method you are calling.**

   E.g.

   ```
   http://endeavour-cim.cloudapp.net/api/v0.1/Organization?identifier=A99999
   ```

2. **Remove the protocol, host, and FHIR 'base' from this URL, according to the following diagram:**



   (FHIR 'base' is specified at the very bottom of this page: http://endeavour-cim.cloudapp.net/)

   E.g. The following URL:

   ```
   http://endeavour-cim.cloudapp.net/api/v0.1/Organization?identifier=A99999
   ```

   Becomes:

   ```
   /Organization?identifier=A99999
   ```

3. **If you are calling a method which has an HTTP request body (POST/PUT methods), then concatenate the URL component created in (2) with the whole request body.**

   E.g. Calling the following URL:

   ```
   http://endeavour-cim.cloudapp.net/api/v0.1/A99999/Slot/1/$book
   ```

   with the following HTTP request body:

   ```
   {
     "resourceType": "Parameters",
     "parameter": [
       {
         "name": "patient",
         "valueString": "11af0e7f-be18-431e-9be9-fd1adb2f0742"
       }
     ]
   }
   ```

   Would result in the following data to hash:

   ```
   /A99999/Slot/1/$book{
     "resourceType": "Parameters",
     "parameter": [
       {
         "name": "patient",
         "valueString": "11af0e7f-be18-431e-9be9-fd1adb2f0742"
       }
     ]
   }
   ```

   (Note: there is no newline or space between the FHIR path and the HTTP request body.)

Example implementations

C#

```
public static string GenerateHash(string fhirPath, string httpBody, string secret)
{
    string dataToHash = fhirPath;

    if (httpBody != null)
        dataToHash += httpBody;

    HMAC mac = HMAC.Create("HmacSHA256");
    ASCIIEncoding encoder = new ASCIIEncoding();
    mac.Key = encoder.GetBytes(secret);
    mac.Initialize();
    byte[] digest = mac.ComputeHash(encoder.GetBytes(dataToHash));

    return Convert.ToBase64String(digest);
}
```

Java

```
private static String generateHash(String fhirPath, String httpBody, String secret) throws
NoSuchAlgorithmException, InvalidKeyException {
    String dataToHash = fhirPath;

    if (httpBody != null)
```

```
        dataToHash += httpBody;

    Mac mac = Mac.getInstance("HmacSHA256");
    SecretKeySpec keySpec = new SecretKeySpec(secret.getBytes(StandardCharsets.UTF_8), "HmacSHA256");
    mac.init(keySpec);
    byte[] digest = mac.doFinal(dataToHash.getBytes(StandardCharsets.UTF_8));
    return new String(Base64.encodeBase64(digest), StandardCharsets.UTF_8);
}
```

JavaScript

```
function generateHash(fhirPath, httpBody, secret) {
    var dataTohash = fhirPath;
    if (httpBody != null)
        dataToHash += body;

    var hash = CryptoJS.HmacSHA256(dataToHash, secret);
    return CryptoJS.enc.Base64.stringify(hash);
}
```

Contact

Please contact jonny.rylands@endeavourhealth.org if you have any queries or issues regarding the above.