PAYE Modernisation

Compressing HTTP Request With GZIP

Version **1.0 Release Candidate 2**

Latest Version History

Version	Element	Change Description
1.0 Release Candidate 2	N/A	Document published

Audience

This document is for any software provider who has chosen to build or update their products to allow for PAYE Modernisation.

Document context

This document provides a non-technical overview of the data items in a payroll submission. This document is designed to be read in conjunction with rest of the Revenue Commissioners' PAYE Modernisation documentation suite including the relevant technical documents.

Overview

The objective of this document is to provide instructions how to compress a HTTP message request for Rest services in standard of paye-employers-webservice-v1 module.

Once the client sends the HTTP message compressed the server will decompress the message to read the content.

Context

Currently Submission & Rates project provides webservice solutions for a variety of different Revenue obligations, these webservice solutions allow for Revenue's customers to comply with their tax or customs obligations.

Client requests are submitted through Soap and Rest webservices but when a request contains a huge body (payload) is necessary to use strategies in server side to keep a good performance of webservice usage.

This document helps how to do compression a HTTP request message and send it an acceptable format to Rest webservices in server side can decompression the message and read it in correct standard. The compression and decompression strategy used for Rest webservices is GZIP.

The GZIP compression is an important way to increase the performance of a Web site and webservices. For some occasions the size reduction of up to 70% lowers the bandwidth capacity needs.

Evaluation

The options considered for the Decompression GZIP in Submission & Rates Webservice project solution is Compression HTTP REST Request.

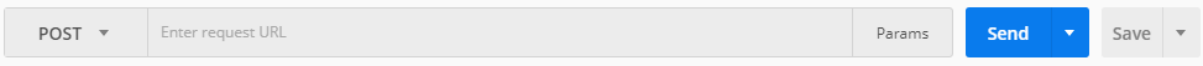
Compression HTTP REST Request:

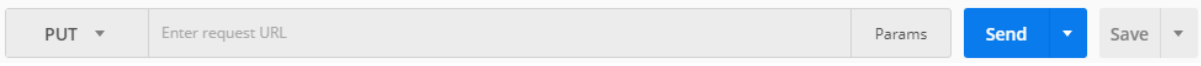
In this approach is necessary to understand the correct usage of HTTP protocol which can spitted in sections below:

1. **Preparing the HTTP request**

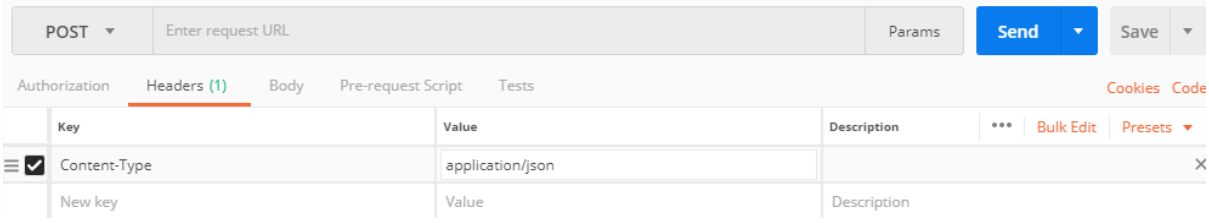
   To prepare HTTP request is important to use the correct Content-Type, Http Method, Content-Transfer-Encoding, a compressed body in GZIP format and encode body request in base64. Please find below the mandatory artifacts to be used:

   - In request use Http Method as "POST" or "PUT"

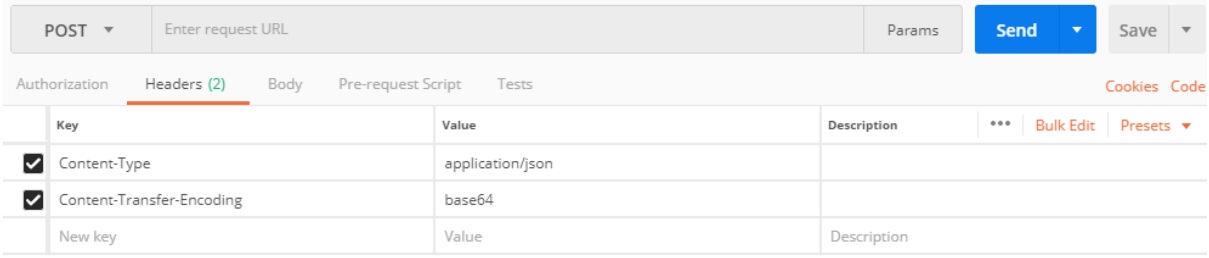




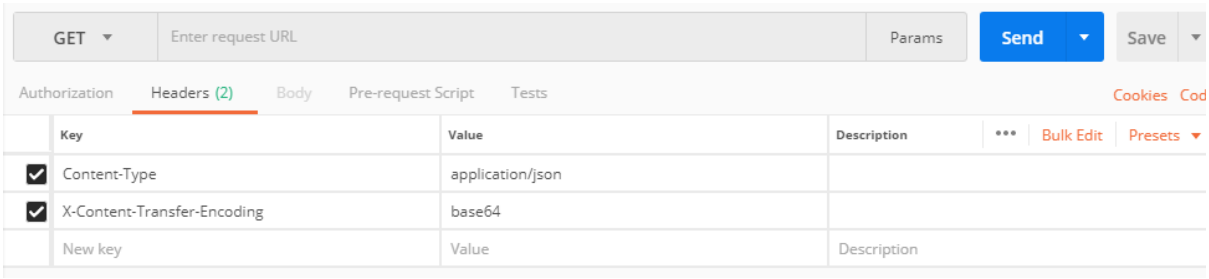
- Set in Headers the Content-Type as “application/json”



- Set in Headers the Content-Transfer-Encoding as “base64”



- An option instead Content-Transfer-Encoding is X-Content-Transfer-Encoding. This occurs due some webservice client don’t sends Content-Transfer-Encoding as default. The correct usage is pick one Content-Transfer-Encoding or X-Content-Transfer-Encoding, please find the correct usage in example below



2. Compressing and encoding the HTTP request body

First is necessary define witch Rest webservice the request will be done, below shows an example of Rest request for webservice Create Temporary RPN.

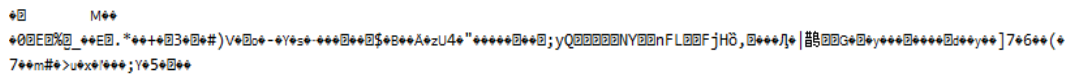
- URL e.g.:

  https://roswebcss-int185/paye-employers/v1/rest/rpn/3390863TH/2018?softwareUsed=abc&softwareVersion=1

- JSON e.g.:

```
{
  "requestId":"86823qwedcxc5677snnnfdsa4551290INT38",
   "newEmployeeDetails": [
      {
         "employeeID": {
            "employeePpsn": "1930368T",
            "employmentID": "1"
         },
         "name": {
            "firstName": "Alice",
            "familyName": "Boyd_TEST"
         }
      }
   ]
}
```


- It’s mandatory to compress the JSON to GZIP format, if you don’t know how to do that please find the section APPENDIX HELPER in the end of this document. Below an example of GZIP



- After you converted the JSON to GZIP is necessary to encode the GZIP to Base64, below an example of Base64 encoded

H4sIAAAAAAAAAE2MywqDMBBFfyXMugtf0eiuRRcuKoW6K6UEM4IQ4yMpVsR/b6wtlFnd
c+fchcCIwxO1yQUkwELm+cOEonpVNIwirZSqheYBpa4XO3lR+gwOBBROWdvLbkZM0fBGa
kjIjSwE8Ivz1JK/fOm1sgQcFkeMBu55m9nLFpX5vIMLZN3WeYu7XTejNsUe4SibCjet5m0j5x
8+dbN4lNm1tLK9O1nfNaQC9dUAAAA=

- Once you followed the recommendations above and already have the headers as Content-Type application/json, Content-Transfer-Encoding base64 and the message body compressed in GZIP and encoded in Base64 your message request is ready to be sent for server.

APPENDIX HELPER

This section shows Java code to help as guide in implementation of GZIP request compression. Please have in mind the development of this solution is up to you and any issue with the code below is not supported by creators of this document.

1. JSON example

```
String payload = "{ \"requestId\":\"86823qwedcxc5677snnnfdsa4551290INT38\", \"newEmployeeDetails\": [ { \"employeeID\": { \"employeePpsn\": \"08978541M\", \"employmentID\": \"1\" }, \"name\": { \"firstName\": \"Alice\", \"familyName\": \"Boyd_TEST\" } } ] }";
```

2. Compressing JSON to GZIP

```
public static byte[] compressJSON(String data) throws IOException {
            ByteArrayOutputStream bos = new
ByteArrayOutputStream(data.length());
            GZIPOutputStream gzip = new GZIPOutputStream(bos);
            gzip.write(data.getBytes());
            gzip.close();
            byte[] compressed = bos.toByteArray();
            bos.close();
            return compressed;

      }
```

3. Encoding GZIP bytes into Base64

```
import org.apache.commons.codec.binary.Base64;

byte[] bytesEncoded = Base64.encodeBase64(data);
```