



PAYE Modernisation

Webservice Compression

Contents

Audience	3
Document context.....	3
Overview	4
Context.....	4
Compressing a REST/JSON Request:.....	4
Preparing the HTTP request	4
Compressing and encoding the HTTP request body	5
Receiving Compressed Responses	7
APPENDIX A.....	8
JSON example	8
Compressing JSON to GZIP	8
Encoding GZIP bytes into Base64	8

Version
Version Date

1.0 Release Candidate 2
28/09/2018

Version History			
Version	Change Date	Section	Change Description
1.0 Release Candidate 2	25/07/2018		New Document
	25/09/2018	Receiving Compressed Responses	Removed compression size limit
	28/09/2018	Compressing a REST/JSON Request	Note added on HTTP method override

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 technical overview of how to receive compressed responses via SOAP or REST and how to compress REST/JSON requests.

Overview

The objective of this document is to provide instructions on how to request compressed responses from the PAYE Modernisation web service and how to compress a REST/JSON request for submission to the PAYE Modernisation web service.

Context

Currently, the web service provides solutions for a variety of different Revenue obligations, these web service solutions allow for Revenue's customers to comply with PAYE Modernisation.

Client requests are submitted through SOAP and REST web services. When either a response or request contains a large body (payload), it may be necessary to compress the payload.

This document describes how to compress a REST/JSON request and send it in an acceptable format to the REST web service. On the server side, the message can be decompressed and read in the correct way. The compression and decompression algorithm used for the REST web service is GZIP. Additionally, this document describes how to send SOAP or REST requests and receive compressed responses.

Compressing a REST/JSON Request:

For this approach it is necessary to understand the correct usage of HTTP protocol which is relayed in the following sections.

Note: We do not accept HTTP method override requests in compressed form, we only accept true POST REST requests in compressed form. The response to the HTTP Method override request will be compressed if the headers are set as described in the ***Receiving Compressed Responses*** section.

Preparing the HTTP request

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

- In the request, use a "POST" HTTP Method.

POST ▾	Enter request URL	Params	Send ▾	Save ▾
--------	-------------------	--------	--------	--------

- In Headers, set the Content-Type to “application/json”

The screenshot shows a REST client interface with a POST method selected. The 'Headers' tab is active, showing a table with one header: 'Content-Type' with the value 'application/json'. The interface includes fields for 'Enter request URL', 'Params', 'Send', and 'Save' buttons. Other tabs like 'Authorization', 'Body', 'Pre-request Script', and 'Tests' are visible.

Key	Value	Description
Content-Type	application/json	
New key	Value	Description

- In Headers, set the Content-Transfer-Encoding to “base64”

The screenshot shows the same REST client interface, but now with two headers: 'Content-Type' (application/json) and 'Content-Transfer-Encoding' (base64). Both headers are checked in the 'Key' column. The interface remains the same with 'POST' method and 'Headers' tab active.

Key	Value	Description
Content-Type	application/json	
Content-Transfer-Encoding	base64	
New key	Value	Description

- Some web service clients don't send Content-Transfer-Encoding by default. To work around this issue we can also accept X-Content-Transfer-Encoding. Please find the correct usage in the example below.

The screenshot shows a REST client interface with a GET method selected. The 'Headers' tab is active, showing two headers: 'Content-Type' (application/json) and 'X-Content-Transfer-Encoding' (base64). Both headers are checked in the 'Key' column. The interface includes fields for 'Enter request URL', 'Params', 'Send', and 'Save' buttons.

Key	Value	Description
Content-Type	application/json	
X-Content-Transfer-Encoding	base64	
New key	Value	Description

Compressing and encoding the HTTP request body

First it is necessary to define on which REST webservice the request will be done. Below shows an example of a REST request for web service New RPN.

- URL e.g.:

<https://softwaretest.ros.ie/pay-employers/v1/rest/rpn/3390863TH/2018?softwareUsed=abc&softwareVersion=1>

- JSON e.g.:

```
{
  "requestId": "86823qwedcxc5677snnnfdsa4551290INT38",
  "newEmployeeDetails": [
    {
      "employeeID": {
        "employeePpsn": "7012602QA",
        "employmentID": "1"
      },
      "name": {
        "firstName": "Alice",
        "familyName": "Boyd "
      }
    }
  ]
}
```

- If a service user is choosing to compress requests it is mandatory to compress the JSON to GZIP format. The section APPENDIX HELPER at the end of this document describes how this is done. Below is an example of GZIP.

M++
00E%0_+eE. *+++@3@2#)V@C+-Y+Z-++@+@S+E+A+zU4+"++++@+@;yQ@@@@N@nFLO@fjH6,@+++/|§@G@+@;++++@++++@+++]7+6+(+
7+m+>uxx'++;Y+5@++

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

H4sIAAAAAAAAAHXNuQqDQBAG4D7gOyxTW3jEI3YGU9hIAuICcNFHWND12A1GJO8e44E
oZqt/+b+Z6ZQDIQRqrF4oZejBA9d2DbNqkCbvxLldR3DOUyrio2XpxkkLo7vpgrMEeDYXPlY
K1rEAGXMMgEeeYzl73VLHDxOOAx6tylX4FoK3hNwNN2wNePmzyd3dl5cDgtBhzX6blaAxz
n+uZyyWsho7MHPWIKgkj0X5yxrZ3guWkq2V5fvFj/KoU9fvG8h820BAAA=

- Once the recommendations above have been adhered to, and the headers have been correctly set and the message body is compressed in GZIP and encoded in Base64, then the message request is ready to be sent for server.

Receiving Compressed Responses

To receive compressed responses from the PAYE Modernisation web services a condition must be met:

- The ***Accept-Encoding*** header must be set to ***gzip***

Requests which omit the Accept-Encoding header will not be compressed.

APPENDIX A

This section shows Java code for the implementation of the GZIP request compression. Please bear in mind that the development of this solution is up to you and any issues with the code below is not supported by creators of this document.

JSON example

```
String payload = "{ \"requestId\": \"86823qwedcxc5677snnnfdsa4551290INT38\",  
  \"newEmployeeDetails\": [ { \"employeeID\": { \"employeePpsn\": \"07012602QA\",  
    \"employmentID\": \"1\" }, \"name\": { \"firstName\": \"Alice\", \"familyName\":  
    \"Boyd_TEST\" } } ] }";
```

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;  
}
```

Encoding GZIP bytes into Base64

```
import org.apache.commons.codec.binary.Base64;  
  
byte[] bytesEncoded = Base64.encodeBase64(data);
```