ICT&L REVENUE

Revenue
Cáin agus Custaim na hÉireann
Irish Tax and Customs

PAYE Modernisation

REST Web Service Integration Guide

Contents

Version **1.0 Release Candidate 2**
Version Date **19/06/2018**

Version History			
Version	**Change Date**	**Section**	**Change Description**
V 1.0 Milestone 1	17/11/2017	All	Document published.
			Audience and **Document Context** sections added.
			Link sources changed
	06/04/2018	4.1.3	Additional information added in relation to digest algorithms used for 'Digest' HTTP header.
1.0 Release Candidate 2	24/05/2018	4.1.3	Additional information added in relation to Content-Type for POST requests.
		2.1.1	HTTP request example Content-Type value changed from 'application/json;charset=UTF-8' to application/x-www-form-urlencoded
	19/06/2018	Payslip	Changed to 'Submission Item'
		4.1.3	(Request-target) updated

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 integrate with Revenue's REST web services including how to sign requests validly. This document is designed to be read in conjunction with the REST/JSON example files as well as the rest of the Revenue Commissioners' PAYE Modernisation documentation suite including the relevant technical documents.

Document References	
Reference	**Document Link**
1. Documents Homepage	Documents Homepage

1. Introduction

This document details the REST PAYE Modernisation web services specification for the following web services:

- Lookup Revenue Payroll Notification (RPN) by Employer web service
- Lookup Revenue Payroll Notification (RPN) by Employee web service
- New RPN web service
- Payroll Submission web service
- Check Payroll Submission web service
- Check Payroll Run web service

The Documents Homepage specified in Document References is the home to all technical documentation, specification, and examples for the above web services which has been made available to enable payroll software developers' to update their software packages to be compatible with PAYE reporting obligations from January 2019. Path locations specified in this document are relative to this Homepage.

This document assumes familiarity with the REST web services above. A full description of each of these can be found in the 'PAYE Modernisation Description Of Web Service Examples' Document under 'PAYE Web Service Examples' on the Documents Homepage.

The OpenAPI Specification, originally known as the Swagger Specification, is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful Web services. The specification can be described using the YAML or JSON format. Revenue provides the specification in JSON format or, alternatively, it can be viewed using the ReDoc UI framework. The JSON file can be downloaded and imported into any Open API UI framework such as the Swagger UI framework. Links are provided for both the JSON format file and the ReDoc UI framework under 'PAYE Web Service Specifications (REST/JSON)' on the Documents Homepage. The URL for each web service will use the HTTPS protocol to ensure the privacy of all communication between ROS and the web service client.

2. Calling the Services

The web services for the PAYE Modernisation messages are described in the REST OpenAPI Specification JSON file provided under 'PAYE Web Service Specifications (REST/JSON)' on the Documents Homepage.

The OpenAPI, REST Security version specifications we are following include:

Open API Specification Version 3.0: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md

HTTP Signatures Version 08: https://tools.ietf.org/html/draft-cavage-http-signatures-08

2.1. REST Endpoints

The PAYE Modernisation web service endpoints are detailed below.

Description	HTTP Method	Endpoint URL	Additional Information	Links
* Look Up RPN By Employer and optionally filter by date last updated and/or employee id's web service	GET	https://www.ros.ie/paye/employers/v1/rest/rpn/{employerRegistrationNumber}/{taxYear}	Query Parameters ○ softwareUsed ○ softwareVersion ○ AgentTain (optional) ○ employeeIDs (optional) ○ dateLastUpdated (optional)	lookUpRPNByEmployer
Lookup RPN by Employee web service	GET	https://www.ros.ie/paye/employers/v1/rest/rpn/{employerRegistrationNumber}/{taxYear}/{employeeId}	Query Parameters ○ softwareUsed ○ softwareVersion ○ AgentTain (optional)	lookUpRPNByEmployee
New RPN web service	POST	https://www.ros.ie/paye/employers/v1/rest/rpn/{employerRegistrationNumber}/{taxYear}	Query Parameters ○ softwareUsed ○ softwareVersion ○ AgentTain (optional)	createNewRPN
Payroll Submission web service	POST	https://www.ros.ie/paye/employers/v1/rest/payroll/{employerRegistrationNumber}/{taxYear}/{payrollRunReference}/{submissionID}	Query Parameters ○ softwareUsed ○ softwareVersion ○ AgentTain (optional)	createPayrollSubmission
Check Payroll Submission web service	GET	https://www.ros.ie/paye/employers/v1/rest/payroll/{employerRegistrationNumber}/{taxYear}/{payrollRunReference}/{submissionID}	Query Parameters ○ softwareUsed ○ softwareVersion ○ AgentTain (optional)	checkPayrollSubmissionComplete
Check Payroll Run web service	GET	https://www.ros.ie/paye/employers/v1/rest/payroll/{employerRegistrationNumber}/{taxYea	Query Parameters ○ softwareUsed ○ softwareVersion ○ AgentTain	checkPayrollRunComplete

		r}/{payrollRunReference}	(optional)	

2.1.1. Additional Information

* If large amounts of employee Id's are being looked up in one request, there is the possibility of URL truncation. In this case integrators should switch from GET to POST, use the *'X-HTTP-Method-Override'* HTTP header and use *'Content-Type=application/x-www-form-urlencoded'* when calling the https://www.ros.ie/ paye/employers/v1/rest/rpn/{employerRegistrationNumber}/{taxYear} endpoint.

Although the *'X-HTTP-Method-Override'* HTTP header is not currently a standard, this is the approach taken by Google for the Google Data API. The X-HTTP-Method-Override header tells the PAYE Modernisation API to treat the request as a GET request, even though it is being conducted as a POST request.

The 'Content-Type' header must be set to 'application/x-www-form-urlencoded' and the request body must meet the 'application/x-www-form-urlencoded' standard. See below HTTP request example

```
POST
v1/rest/rpn/0000001W/2019/1/1?agentTain=11221w&softwareUsed=SoftwareXYZ&sof
twareVersion=1.0 HTTP/1.1
Host: www.ros.ie
Date: Wed Oct 04 16:35:51 BST 2017
Content-Type: application/x-www-form-urlencoded
X-HTTP-Method-Override: GET

// Request Body
employeeIDs={employmentID-1}&employeeIDs={employmentID-2}&employeeIDs={
employmentID-3}
```

2.2. Digital Signatures

The PAYE Modernisation web services will require a digital signature. This will be the digital signature of the declarant.

3. Interpreting the Responses

Each web service will return a response message to the client as outlined below.

3.1. Validation Errors

3.1.1. Request Validation

When a request is made to a PAYE Modernisation web service three checks are carried out before any processing can occur. These include:

1. Authentication
2. Authorisation

3. JSON API validation

Step 1 of the validation process tries to verify the authenticity of the message by checking it has been signed with a valid digital signature. Step 2 focuses on authorising the credentials and finally, step 3 checks the message is valid using JSON API validation.

If there are any errors encountered during the above processes, the response will include a HTTP status code of either 401 (Unauthorised) if authentication fails, 403 if authorisation fails and 400 (Bad Request) if JSON API validation fails. The message body will provide more information on the details of the problem. Where an error code is returned to the client, no processing will occur for that message.

If the message passes JSON API validation, authentication, and authorisation but the resource cannot be found the response will include a HTTP status code of 404. If the resources query parameters are not as required or are missing, the response will include a HTTP status code of 400. Where an error code is returned to the client, no processing will occur for that message.

3.1.2. Line Item Validation

Once the message passes JSON API validation, authentication, and authorisation, lower level line item validation is carried out on the following requests:

- Lookup RPN
- New RPN
- Payroll Submission Request

A successful response is sent back to the client, detailing any validation errors that occurred on the request. The code i.e. the technical error code used for mapping to the error message, the path to the error in the schema and the description of the error is detailed in the response.

A list of all validation rules carried out for each request can be found in the Validation Rules document on the Documents Homepage under 'Supporting Documentation'.

3.2. Lookup Revenue Payroll Notification (RPN) Web Service

The "Lookup RPN by Employer", "Lookup RPN by Employee" and "Lookup RPN using multiple Employee Id's" web service Response will return the most up to date RPN details for employees listed in the request, if the employee has an RPN associated with the employer. Employees that do not have an RPN associated with the employer are returned in the response with no RPN details. A New RPN needs to be requested for these employees using the New RPN Request Web Service.

A list of validation errors (if any) on the Lookup RPN Request is also included in the response. Please refer back to Section 3.1 for more information on Validation Errors.

3.3. New RPN Web Service

The Employer's New RPN response will return new RPN details for the employees requested. PPSN and Employment ID of employees are returned with no RPN details where new RPN details could not be created.

A list of validation errors (if any) on the New RPN Request is also included in the response.

3.4. Payroll Submission Web Service

The Payroll Submission response returns an acknowledgement status for the Employer's PAYE Payroll Submission Request.

If validation failed a list of the Submission validation errors are returned in this response.

3.5. Check Payroll Submission Web Service

Check Payroll Submission will return the current status of an employer's PAYE payroll submission. The possible status values are Pending or Completed.

If the status is completed, the response includes summary totals of valid submission items. If the status is pending the response does not contain any summary totals. Validation errors for any invalid submission items are listed as well as any validation errors on the Check Payroll Submission Request. Invalid submission items are not saved therefore their amounts do not feed into the employer liability.

3.6. Check Payroll Run Web Service

Check Payroll Run will return the current status of an Employer's PAYE payroll run. The possible status values are Pending or Processed. If any submissions making up the payroll run are at a status of Pending then the status of the payroll run response will also be at Pending.

If the status is Processed the response includes a list of submissions that make up the payroll run and includes summary details of all processed submissions.

A list of validation errors (if any) on the Check Payroll Run Request is also included in the response.

4. Digital Signatures

Any ROS web service request that either returns confidential information or accepts submission of information must be digitally signed. This must be done using a digital certificate that has been previously retrieved from ROS.

The digital signature must be applied to the message in accordance with the HTTP Signatures specification as specified in Section 4.1.

The digital signature ensures the integrity of the document. By signing the document we can ensure that no malicious intruder has altered the document in any way. It can also be used for non-repudiation purposes.

If a valid digital signature is not attached, a HTTP status code of 401 (Unauthorised) will be returned. The message body will provide more information on the details of the problem.

4.1. HTTP Signatures

The HTTP signatures protocol is intended to provide a simple and standard way for clients to sign HTTP requests. A summary of the structure of a HTTP Signature is outlined below. This is a simplified explanation of the HTTP Signatures specification. The full specification can be found at Signing HTTP Messages and should be read in full. The specification defines two approaches to building a HTTP signature, "The 'Signature' HTTP Authentication Scheme" and "The 'Signature' HTTP Header", Revenue uses the latter.

At a high level, a HTTP Signature is a HTTP header that is added to a HTTP request. It is comprised of a set of components that were used to generate a digital signature and the digital signature itself.

4.1.1. HTTP Signature Sample

Below is a sample HTTP Signature header.

```
Signature: keyId="MIICfzCCAeigAwIBAgIJ... // truncated",
algorithm="rsa-sha512",
headers="(request-target) host date digest",
signature="GdUqDgy94Z8mSYUjr/rL6qrLX/jmudS... // truncated"
```

4.1.2. HTTP Signature Components

The Signature HTTP header contains four components, keyId, algorithm, headers and signature. Below is a description of each.

keyId: The keyId field must contain a Base64 encoded version of the X509 certificate that accompanies the private key used to sign the message. This field is required.

algorithm: The `algorithm` parameter is used to specify the digital signature algorithm to use when generating the signature. Revenue expects this to be 'rsa-sha512'. This field is required.

headers: The `headers` parameter specifies the list of headers used when generating the signature for the message. The parameter must be a lowercased, quoted list of HTTP header fields, separated by a single space character. The list order is important, and MUST be specified in the order the HTTP header field-value pairs are concatenated together during signing.

signature: The signature component is a base 64 encoded digital signature. The implementer uses the `algorithm` and `headers` field to form a canonicalized `signing string`. This `signing string` is then signed with the private key that accompanies the X509 certificate associated with the `keyId` field and the algorithm corresponding to the `algorithm` field. The `signature` field is then base 64 encoded.

4.1.3. Signature String Construction

In order to generate the string to be signed, the implementer MUST use the values of each HTTP header defined in the `headers` signature field, to build the signature string. Values must be in the order they appear in the `headers` signature field. If the associated HTTP header does not exist, it should be added to the HTTP request BEFORE attempting to construct this string.

Allowable values in the headers field are outlined in the table below.

Value	Mandatory
* (request-target)	Yes
host	Yes
date	Yes
** x-date	Yes, if date header cannot be added.
*** digest	Yes, if HTTP method is of type POST
**** content-type	No
content-length	No
x-http-method-override	If HTTP method is of type POST, HTTP header *'X-HTTP-Method-Override'* exists and *'Content-Type=application/x-www-form-urlencoded'*. See Section 2.1.1 for more detail.

* The `(request-target)` header field is a special header field in that its value is comprised of 2 HTTP headers. It is generated by concatenating the lowercase HTTP method, an ASCII space, and the request path headers. See below for sample

(request-target): post /paye-employers/v1/rest/rpn/{ employerRegistrationNumber }/{taxYear}?softwareused=XYZ&softwareVersion=1.0

** The 'x-date' headers field value should ONLY be used in conjunction with the X-Date HTTP header if a Date HTTP header cannot be added to the HTTP request programmatically. The Date header has a limitation when using javascript in a browser to build and send a HTTP signature. The limitation is that you cannot add a 'Date' HTTP header when executing javascript in a browser. The native xmlhttprequest object prohibits addition of a 'Date' HTTP header. Building the signature string that will be signed with an 'x-date' header instead of a 'date' header removes this restriction.

*** The 'Digest' HTTP header is created using the POST body/payload. The payload should be converted to a byte array, hashed using the SHA-512 algorithm and finally base64 encoded before adding it as a HTTP header.

**** Although Content-Type does not have to be part of the Signature string, it is required as a HTTP Header if HTTP Method Type is POST. If the request is a standard POST (i.e. not a header '*X-HTTP-Method-Override*'POST, then the Content-Type should be set to 'application/json' or 'application/json;charset=UTF-8'

All other header field values are created by concatenating the lowercase header field name followed by an ASCII colon `:`, an ASCII space ` `, and the header field value. Leading and trailing whitespace in the header field value MUST be omitted. If the header field is not the last value defined in the `headers` signature field, then append an ASCII newline `\n`

4.1.4. Signature Creation

The signature component is a base 64 encoded digital signature. The implementer uses the `algorithm` and constructed Signature String. The Signature String is signed with the private key that accompanies the X509 certificate associated with the `keyId` field and the algorithm corresponding to the `algorithm` field. The `signature` field is then base 64 encoded.

5. Example Messages

There is an adjoining file 'REST_Web_Service_Integration_Guide_Sample_Http_Messages.txt' that contains HTTP Request/Response samples describing the structure of the request and outlining the possible responses a client can expect. Monetary figures in all examples are for illustrative purposes only.

Appendix A – Extracting from a .p12 File

Each customer of ROS will have a digital certificate and private key stored in an industry standard PKCS#12 file.

In order to create a digital signature, the private key of the customer must be accessed. A password is required to retrieve the private key from the P12 file. This password can be obtained by prompting the user for their password.

The password on the P12 is not the same as the password entered by the customer. It is in fact the MD5 hash of that password, followed by the Base64-encoding of the resultant bytes.

To calculate the hashed password, follow these steps:

1. First get the bytes of the original password, assuming a "Latin-1" encoding. For the password "Password123", these bytes are: 80 97 115 115 119 111 114 100 49 50 51(i.e. the value of "P" is 80, "a" is 97, etc.).
2. Then get the MD5 hash of these bytes. MD5 is a standard, public algorithm. Once again, for the password "Password123" these bytes work out as: 66 -9 73 -83 -25 -7 -31 -107 -65 71 95 55 -92 76 -81 -53.
3. Finally, create the new password by Base64-encoding the bytes from the previous step. For example, the password, "Password123" this is “QvdJref54ZW/R183pEyvyw==”.

This new password can then be used to open a standard ROS P12 file.