**Jupiter Workflow**

**Technical Guide for Jupiter Job Submission**

**Christopher Hall**

**Technical Director**

**June 2020**

[Overview](#_Toc42082083)

[Job Submission pseudo-code](#_Toc42082084)

[Reconciliation File](#_Toc42082085)

[Zip File Creation](#_Toc42082086)

[MD5 Sum Creation](#_Toc42082087)

[Access to AWS S3 Bucket](#_Toc42082088)

[Amazon Client Classes](#_Toc42082089)

[Deliver Zip File for Processing](#_Toc42082090)

[Generate Delivery XML](#_Toc42082091)

[Post to SQS](#_Toc42082092)

[Subsequently](#_Toc42082093)

[Error Response](#_Toc42082094)

## Overview

This document details how account holders can integrate programmatically with Jupiter without installation of any software being provided by us.

All of the following steps are achieved in one or two API calls that we have pre-defined, and our preference is for customers to make use of our .NET or Java API, or otherwise use Command Line interfaces to achieve this.

### Job Submission pseudo-code

The pseudo-code listed below covers the processing needed at the customer site to prepare for and submit items using the Jupiter exposed, and Amazon AWS REST based web services.

* Create reconciliation file called “dtp-rec.csv” to itemise the content transferred
* Create Zip file with the transfer directory contents in temporary storage
* Obtain MD5 hash sum of zip file
* Initialise Access to Amazon S3 location via Web Service call to Jupiter
* Transfer Zip file to S3 bucket
* Generate Delivery XML document and write to Amazon SQS
* Remove zip file
* End

The remaining sections of this document cover all aspects of these pseudo-code points.

### Reconciliation File

This file should be called “dtp-rec.csv” and it must itemise all documents in a single directory to be transmitted. The comma-delimited CSV content should include:

|  |  |
| --- | --- |
| **Column Name** | **Content** |
| DocName | File Name - Case sensitive |
| Country | ISO3166 Alpha-2 Country Code for destination |
| Postcode | Postcode of destination – if available |
| Copies | Copy count this file |

* Reconciliation file Content should include reference to itself and the file should be written to the transfer directory
* The case sensitive File Names you create should contain some reference to the contained data. Eg. Account/Invoice Number so that they can be searched for and retrieved in the Jupiter Web portal
* If required by arrangement, Jupiter can associate additional reporting 'meta-data' with each document in our database by detailing extra columns in this file

### Zip File Creation

The created zip file name must conform to the convention:

<jupiter user>-yymmdd-hhmmss.zip

* The <jupiter user> value will be supplied to you once your account is created during the on-boarding process.
* The zipped file content must be at the 'root' of the zip, with no directory path information stored
* Use a standard compression algorithm exposed through use of standard libraries supllied with .NET or Java Zip classes
* Write the zip file to temporary storage for subsequent deletion after transfer.

### MD5 Sum Creation

We use MD5 sum values to verify that the content that you send is unchanged and complete at the point of our processing on the Jupiter servers. If your transferred content does not match the values you tell us, then the job is rejected.

You can obtain MD5 sum values using libraries such as System.Security.Cryptography.MD5 in .NET or DigestUtils.md5Hex from apache commons codec in Java.

### Access to AWS S3 Bucket

You need to gain write access to the Jupiter inbound data store in order to make your zipped content available for processing. This is achieved through the retrieval of a temporary Access Key Identifier and Secret Key that we expose to you via a web service.

This is a standard REST based web service implementation and you can construct your process using an endpoint URL of:

https://<implementation>/WorldShip/rs/S3AccessResponder?restletMethod=get

Note HTTPS ONLY is supported by this service. The process requires a HTTP header containing

* HTTP Basic Authentication (base 64 encoded) parameters added which incorporate the Jupiter User Id (not access key) and password (not secret key) that we supply during the on-boarding process

The response is an XML document identified by the S3Access.xsd schema defined below.

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<xs:schema attributeFormDefault=*"unqualified"*

elementFormDefault=*"qualified"* targetNamespace=*"x-schema:AccessDocument.xsd"*

xmlns=*"x-schema:AccessDocument.xsd"*

xmlns:xs=*"http://www.w3.org/2001/XMLSchema"*>

<xs:complexType name=*"queueIdentifier"*>

<xs:simpleContent>

<xs:extension base=*"xs:string"*>

<xs:attribute name=*"queueid"* type=*"xs:int"* />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:element name=*"s3access"*>

<xs:complexType>

<xs:sequence>

<xs:element type=*"xs:string"* name=*"user"* />

<xs:element type=*"xs:string"* name=*"keyid"* />

<xs:element type=*"xs:string"* name=*"secretkey"* />

<xs:element type=*"xs:string"* name=*"bucket"* />

<xs:element type=*"queueIdentifier"* name=*"postqueue"* />

<xs:element type=*"xs:string"* name=*"origin"* />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

The returned access key details, bucket, and post queue values provided should be used in the AWS SDK method calls identified later.

Note that these values are subject to change at any time so once retrieved they should be held only transiently for the purpose of a single transfer. The values may be different on your next retrieval.

### Amazon Client Classes

Use the Amazon AWS SDK available at <http://aws.amazon.com/tools/>

Each of the AWS SDK calls identified below make use of the respective Amazon Client classes. Eg. AmazonS3 and AmazonSQSClient. In each case the classes and calls listed are recommended mechanisms for the transfer, but the SDK provides for multiple means of achieving the same thing.

Developers are free to use their own approach, so long as the requirements here are complied with. Refer to the Amazon documentation for example uses of the class items listed below in conjunction with these client classes.

### Deliver Zip File for Processing

Once the Zip file has been created and you have calculated an MD5 sum value of that file, you should transfer the file to the AWS S3 bucket identified in the S3Access XML.

Transfer the file using the AmazonS3, PutObjectRequest and TransferManager classes.

Ensure the following meta data attributes are included via adding an ObjectMetaData class instance to the PutObjectRequest:

* MD5Sum
* Mime ContentType

### Generate Delivery XML

A successful response from writing the Zip file to S3 should be followed by the posting of an XML document to AWS SQS to trigger Jupiter processing. A sample of the minimum XML content required follows:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<DELIVERY>

<VERSION>16</VERSION>

<DESCRIPTION>User description of this job</DESCRIPTION>

<JOBCODE>JOB\_CODE\_ALLOCATED</JOBCODE>

<ZIPFILEPARTS MD5=*"abc123"* bucket=*"bucketname"*>

<ZIPPART>jupiterUser-130507-223054.zip</ZIPPART>

</ZIPFILEPARTS>

</DELIVERY>

During the on-boarding process we will discuss your Jupiter processing requirements and the JOBCODE node value will be supplied to you based on your specific processing and distribution needs.

Note that the ZIPFILEPARTS node contains both the MD5 attribute you calculated earlier as well as the bucket node contained in the retrieved S3Access XML.

### Post to SQS

Once the Delivery XML has been created you should create an MD5 sum of the serialized document and write the XML to AWS SQS using the SendMessageRequest and AmazonSQS classes.

Verify success of the message posting by comparing your XML MD5 calculation with the response from the post.

### Subsequently

Once your transfer and message posting has completed successfully Jupiter will automatically process the inbound data and allocate print files to one or more locations based on the Country code you supplied for each item and our agreed distribution profile.

You can monitor the throughput of your jobs manually via a web portal using a URL that we create during the on-boarding process for you. Alternatively you can use a further web service to query the status of transmitted items for use programmatically in your application. Let us know if this is of further interest.

### Error Response

When errors are encountered in the Jupiter web services – then an HTTP 500 error is returned in all cases with comments in the response as to the cause of the error.

Refer to AWS documentation for how to deal with errors encountered in each call made.