**Jupiter**

**Integration via AWS tools**

**Functional Specification v2.0**

**Christopher Hall**

**Technical Director**

**June 2020**

[Executive Summary](#_Toc42079662)

[Process Overview](#_Toc42079663)

[Print Data Reconciliation file](#_Toc42079664)

[Create a zip archive of all data](#_Toc42079665)

[Upload to S3](#_Toc42079666)

[Delivery XML document](#_Toc42079667)

[Sample Code](#_Toc42079668)

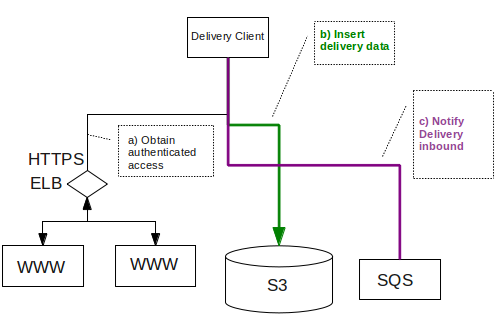
## Executive Summary

The Jupiter application software makes full use of the HTTPS protocol to implement a transfer mechanism which fulfills the need for a secure, validated, automated and/or optionally manual transfer mechanism to create jobs in Jupiter for global distributed print.

The integration processing outlined in this document provides details of the low-level calls that can be implemented in a calling application. This process effectively by-passes the need to integrate any supplied application binaries, but instead allows the use of publicly available Amazon APIs in the programming language of preference by each integrator.

## Process Overview

* Create a Print Data Reconciliation file and place it with the print data in a single directory
* Zip the content of the local directory for transfer to Jupiter, and obtain an MD5 hash signature of that file
* Use the provided access key id and secret access key to write to the Amazon Cloud Simple Storage Solution (S3) via HTTPS to create Jupiter job data
* Create an XML document (including the zip file MD5 hash) and write it to the Amazon Simple Queue Service (SQS) message facility to commence Jupiter processing



### Print Data Reconciliation file

The print data must be accompanied by a print data CSV reconciliation file – which is always to be called ‘dtp-rec.csv’

The reconciliation file is the mechanism by which transferred files are identified to the Jupiter Application, and through which routing decisions and / or additional document meta-data can be associated with documents.

As a minimum, four mandatory CSV columns are needed:

DocName,Country,Postcode,Copies

Where required by pre-agreement, this file content can be extended to store any number of data items referred to as meta-data to be associated with individual documents in the Jupiter database. For Instance, these extra columns can be the address elements that relate to an individual document.

All files (whether for printing or not) are to be detailed in this file and this file should be placed in the same directory as the print data, or otherwise added to the same zip file as the transferred print data.

### Create a zip archive of all data

Any programming tool available on the integrators platform can be used to create a single zip file containing all content to be transferred to Jupiter. Content should be in the root of the zip, and not contained in folders within that file.

Typically this takes the form of using an API such as the .NET ‘SharpZipLib’ or the Java zip libraries to compress the data to a single archive for transfer.

The created zip file must conform to a file naming convention which identifies the customer creating the transfer to Jupiter Processing. The naming convention is as follows:

<JupiterUserId>-yymmdd-hhmmss.zip

and once created, the file must be parsed by an MD5 algorithm to identify its hashed value. This zip file name value is then passed to Jupiter in the XML document identified below.

JupiterUserId here does not refer to the AWS Access credentials, but will be supplied to you separately to the Access Key Id.

### Upload to S3

The zipped file must be transferred to Amazon S3 using the access credentials provided by us. Specifically, the file should be written to an S3 Bucket using the available SDK transfer methods of either the PutObjectRequest or the Upload class.

AWSEndPoint: s3.amazonaws.com

AWSBucketName: pb-us2-clientinboundbucket

Note that Write access only is granted for the supplied credentials, once transferred the file cannot be read or deleted without being processed by Jupiter back office processes. The file will only be processed if the MD5 Hash is found to be the same as informed via the XML SQS message as detailed below.

### Delivery XML document

For Jupiter to process the uploaded content, an XML document must be published to the AWS SQS message queue facility using the SQS SendMessageRequest class to the following URL:

https://sqs.us-east-2.amazonaws.com/375228553146/pb-us2-OrbitDelivery

Note that in each case the correct AWS Region relevant to your implementation must be supplied to the SDK client object to ensure correct communication transport.

Usage of the AWS SendMessageRequest API includes reading the composed XML into a String object and sending that String. Integrators will wish to verify the success of the transfer, by referring to and comparing the API response which is an MD5 hash of the sent content.

The XML document itself must conform to the following XSD:

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

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="DELIVERY" type="DELIVERYType"/>

<xs:complexType name="PROPERTYType">

<xs:simpleContent>

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

<xs:attribute type="xs:string" name="name" use="optional"/>

<xs:attribute type="xs:string" name="value" use="optional"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:complexType name="PROPERTIESType">

<xs:sequence>

<xs:element type="PROPERTYType" name="PROPERTY" maxOccurs="unbounded" minOccurs="0"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="ZIPFILEPARTSType">

<xs:sequence>

<xs:element type="xs:string" name="ZIPPART"/>

</xs:sequence>

<xs:attribute type="xs:string" name="MD5"/>

<xs:attribute type="xs:string" name="bucket"/>

</xs:complexType>

<xs:complexType name="DELIVERYType">

<xs:sequence>

<xs:element type="xs:byte" name="VERSION"/>

<xs:element type="xs:string" name="DESCRIPTION"/>

<xs:element type="xs:string" name="JOBCODE"/>

<xs:element type="PROPERTIESType" name="PROPERTIES"/>

<xs:element type="ZIPFILEPARTSType" name="ZIPFILEPARTS"/>

</xs:sequence>

</xs:complexType>

</xs:schema>

Notes:

* This is a partial schema which can be used as a bare minimum to transfer jobs into the Jupiter processing
* Version should be specified as 16
* Description is the user contextual value to appear on the Jupiter website describing this job
* JOBCODE is the value allocated by your onboarding team to drive the customer/job specific process that occurs on the Jupiter server, relating to data processing and print centre distribution. The value ‘PA004’ can be used as a default, which relates to the standard - non specialised - processing of fully composed PDF files.
* Zero or more PROPERTY nodes can be added to further tailor Jupiter processing. A wide range of name/value pairs can be specified here under onboarders instruction.

For instance, name=”PN\_STATUS\_AFTER\_PROCESSING” value=”-1” indicates that the job should be placed in a Booked status for online approval before distributing for print

* A single ZIPPART node must be added which contains the name of the zip file that was uploaded to S3, and its two attributes must be present as they relate to the uploaded data. Note that the MD5 Sum value must be a 32 character Hexadecimal String, and not the Base64 encoded value that the PutObjectRequest class requires for the S3 upload.

## Sample Code

Our sample code demonstrates the delivery, job download and transaction log download. Configuration is done via the App.config file in each project.

**JupiterDelivery**

This implements the steps for a print customer adding jobs to the Jupiter Workflow Management Console. It implements the CSV file and Zip file creation, Web Service calls to Amazon and housekeeping actions.

|  |  |
| --- | --- |
| **Key** | **Description** |
| amazonRegionEndpoint | Amazon region endpoint allocated by Jupiter (e.g. eu-west-2, us-east-1 for full list see <https://docs.aws.amazon.com/general/latest/gr/rande.html>). |
| jupiterServer | Server URL allocated by Jupiter. |
| clientDirectory | Directory containing PDF files for print (Eg. C:\Work\pdf\ /var/work/pdf/). |
| deliveryDescription | Descriptive name for the job (Eg. "Test Delivery", can be unique for each job). |
| jobCode | Pre-agreed code allocated by Jupiter Onboarding team. |
| userName | Username of an account in the Jupiter Workflow Management System. |
| password | Password for the above user. |

**JupiterDownload**

This implements the steps to for a printer download any new jobs for printing showing the connection to Amazon service, download of documents and Web Service calls to update the status for each.

|  |  |
| --- | --- |
| **Key** | **Description** |
| amazonRegionEndpoint | Amazon region endpoint allocated by Jupiter (e.g. eu-west-2, us-east-1 for full list see <https://docs.aws.amazon.com/general/latest/gr/rande.html>). |
| jupiterServer | Server URL allocated by Jupiter Onboarding team. |
| clientDirectory | Directory for download of PDF files for print (Eg. C:\Work\pdf\ /var/work/pdf/). |
| userName | Username of an account in the Jupiter Workflow Management System. |
| password | Password for the above user. |

**JupiterTransactionLog**

This implements the steps to download the latest logs of all transactions on the Jupiter Workflow Management Console.

|  |  |
| --- | --- |
| **Key** | **Description** |
| amazonRegionEndpoint | Amazon region endpoint allocated by Jupiter (e.g. eu-west-2, us-east-1 for full list see <https://docs.aws.amazon.com/general/latest/gr/rande.html>). |
| jupiterServer | Server URL allocated by Jupiter. |
| userName | Username of an account in the Jupiter Workflow Management System. |
| password | Password for the above user. |

**JupiterImplementation**

Application code for the above three projects.