# OaaS: Optimization as a Service

***Fabio Giannetti; Hewlett-Packard Laboratories; Palo Alto, CA/USA***

## Abstract

*In any flexible manufacturing environment, like a Print Service Provider (PSP), there are fixed cost, related to job handling and machine set-up/cleaning. The Print Production is a complex environment where resources, devices and print jobs need to be coordinate and optimized to increase productivity and/or reduce cost.*

*The inefficiencies can be significantly reduced, if production is analyzed as a whole and jobs are subject to optimization steps that streamline their production. The most common optimizations that can be performed are:*

- *Planning. This allows identifying the required steps to produce the job, given the set of currently available devices.*
- *Ganging. This operation physically places more than one job on a printed sheet or roll, enabling to significantly reduce job handling costs, paper waste and, if applicable mailing costs.*
- *Batching. This operation serializes the order in which several jobs are processed. This tends to minimize the set-up and cleaning time at various devices, increasing efficiency.*

*The proposed approach eliminates the need of having such optimization components installed at the PSP premises providing a Pay-as-You-Go solution [1] as well as leveraging 3rd parties data and capabilities.*

## Introduction

Print Service Providers (PSPs) are usually categorized as flexible manufacturing due to the jobs and processes diversity that are utilized in production. In such environments, inefficiency is hard to identify and eliminate. The variety of jobs and processes significantly increases the amount of set-up and clean-up for all the machines involved and requires several production paths that can be interfering with each other and cause delays.

The marketplace offers various solutions and products that can improve the production of different job categories. Each solution is typically tailored to a job category of family and tends to produce significant benefits only within that family boundary, e.g., photo applications vs. flyers vs. postcards.

If a PSP has any specific application as a mainstream, then it is a good investment to acquire solutions that streamline production. But if these jobs are occasional or not predictable in the future, it may be too costly to optimize for marginal orders.

A further aspect is that once a solution has been acquired and put in place, it may not be able to maintain the same level of efficiency introduced during the evolution of the PSP capabilities. For instance, when new presses or finishing equipment are acquired, there could be better ways of improving job production, like better imposition, automated job inspection, and calibration.

## Proposed Approach

The proposed approach leverages a centralized cloud based system, which exposes several optimization services [2]. These services are not deployed to the PSP, but rather used to optimize jobs that are going into production.

These services can be called to plan production for the entire day, shift or to simplify production of specific types that are less common and/or more speculative.

The Optimization as a Service (OaaS) platform is not a Web-to-Print substitute and it is not designed to help PSPs acquire jobs. The OaaS is instead a platform that receives a set of jobs and returns various optimization strategies back to its caller.

For efficiency reasons, the platform must operate on jobs descriptions and not have to deal directly with artwork or assets.

In this way the amount of data sent to the cloud is usually comprised of eXtensible Markup Language (XML) [3] based content and it can be highly compressed and moved efficiently.

Since there are already existing standards that can be leveraged, we are planning to design our platform around these.

In particular we believe that the combination of the Job Definition Format (JDF) and the Job Messaging Format (JMF) [4] is expressive enough to satisfy the data required by these services.

The platform acts as a Software as a Service (SaaS) system. Remote clients can send requests to the system, which returns the result of applying a service to the input data. The requestor will send a request to process a job (or job set). The OaaS Service Host validates the jobs, then acknowledges the request and releases the requestor. The next step is to route the jobs to the appropriate Optimizer. Once the Optimizer has performed its task, the controller returns the optimized job(s) to the requestor, as illustrated in ***Figure 1***.

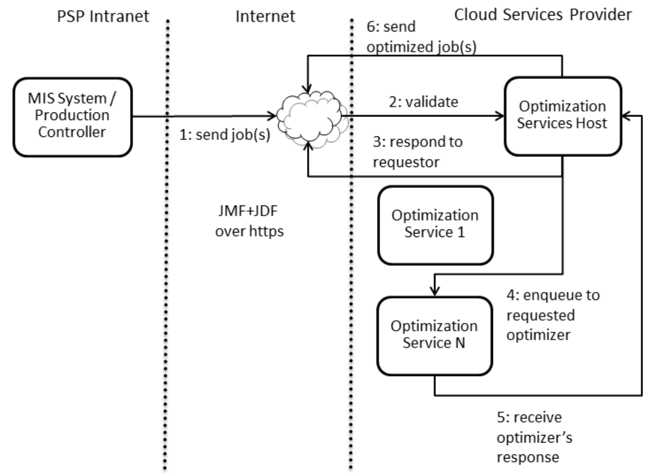


***Figure 1:*** *Submitting Job(s) to OaaS*

In the next sub-sections we will discuss the communication and routing protocol (discover ⇒ request ⇒ response) as well as the vendor's driven addition of device specifications to the system.

## *Communication and Routing Protocol*

The first operation that an OaaS client will execute is to identify all the available services that the platform offers. We call this phase discover. This approach is similar to the request of known devices for a subsystem. JMF supports a specific query KnownDevices, which we are leveraging for the discovery part of our communication protocol as illustrated in Excerpt 1.

```
<Query ID="QDevs" Type="KnownDevices" SenderID="MIS"
xsi:type="QueryKnownDevices"/>
```

***Excerpt 1:*** *JMF's Known Devices Query*

Once this command is executed, the OaaS Service Host will respond sending all the optimization services that are currently available along with their description, as illustrated in ***Excerpt 2***.

```
<Response ID="RDevs" refID="QDevs" Type="KnownDevices"
xsi:type="ResponseKnownDevices">
  <DeviceList>
     <DeviceInfo DeviceStatus="Idle">
        <Device DeviceID="OaaS:Planning" DeviceType="Planning"/>
     </DeviceInfo>
     <DeviceInfo DeviceStatus="Idle">
        <Device DeviceID="OaaS:Ganging" DeviceType="Ganging"/>
     </DeviceInfo>
     <DeviceInfo DeviceStatus="Idle">
        <Device DeviceID="OaaS:Batching" DeviceType="Batching"/>
     </DeviceInfo>
  </DeviceList>
</Response>
```

***Excerpt 2:*** *JMF's Response for KnownDevices*

The OaaS client can now submit multiple job requests. The job requests that need to be addressed altogether, i.e., Ganging or Batching operations can be linked using a **GangName** and **GangPolicy** strategy. In this way jobs that are submitted under this strategy are processed only when the entire set is submitted. The last command called ForceGang will tell the device to proceed and process the set, as illustrated in ***Excerpt 3***.

```
<Command ID="QE_1" Type="SubmitQueueEntry"
xsi:type="CommandSubmitQueueEntry">
   <QueueSubmissionParams GangName="Test" GangPolicy="Gang"
URL="file://NetworkShare/Jobs/Job1.jdf"/>
</Command>
…
<Command ID="QE_N" Type="SubmitQueueEntry"
xsi:type="CommandSubmitQueueEntry">
  <QueueSubmissionParams GangName="Test" GangPolicy="Gang"
URL="file://NetworkShare/Jobs/JobN.jdf"/>
</Command>
<Command ID="GNG" Type="ForceGang" xsi:type="Command
ForceGang">
   <GangCmdFilter GangName="Test"/>
</Command>
```

***Excerpt 3:*** *Job Submissions as a Set*

The second aspect is how to enable job's routing within the OaaS. In the ***Excerpt 2*** we were able to collect from the OaaS controller all the available devices, this enables the service caller to specify what device the jobs should be routed to.

Once the OaaS Service Host receives the job(s) it identifies the appropriate optimizer checking the **GeneralID** content value.

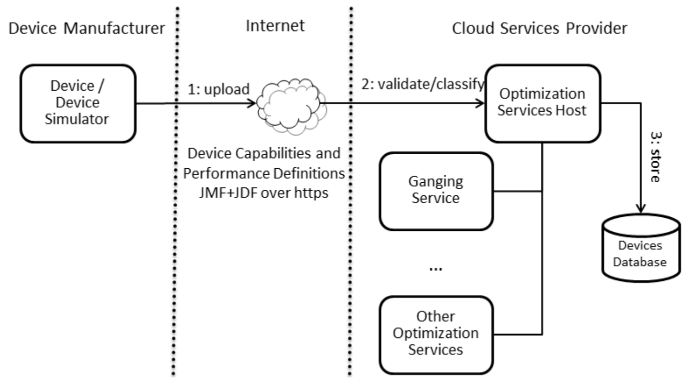
```
<JMF SenderID="MIS" MaxVersion="1.4"
xmnls="http://www.CIP4.org/JDFSchema_1_1">
   <GeneralID IDUsage="OaaS:Routing" IDValue="OaaS:Planning"/>
   <Command ID="QE_1" …> … </Command>
   …
   <Command ID="QE_N" …> … </Command>
   <Command ID="GNG" …> … </Command>
</JMF>
```

***Excerpt 4:*** *Routing based on GeneralID*

Each of the Optimizers is handled as a JDF Device. This means that the Service Host can queue the jobs to the corresponding Optimizer's queue and act as a JDF controller.

## *Adding Device Specifications*

Many Optimizers rely on device specifications in order to perform optimizations. One of the biggest advantages of the OaaS is the ability of dynamically grow the supported devices and their characteristics, having the manufacturers contributing to the knowledge base. This has mutual advantages in the system. The Optimizers developers can now offer visibility to the latest devices and their capabilities without having to refresh their applications or send patches or updates to all PSPs that use the application. Device manufacturers can make sure that their latest device is available and correctly represented. This allows taking full advantage of that device in any optimization task. PSPs have an up to date repository that will optimize their jobs for the set of the devices that they currently have in their floor. The only thing that PSPs have to do is to send manufacturer name, device model and review of the devices they currently own and the Optimizers can access to all the data directly in the cloud. This also removes potential errors and misrepresentation of the devices and or theirs characteristics.



Device Manufacturer
Internet
Cloud Services Provider
Device / Device Simulator
1: upload
2: validate/classify
Optimization Services Host
3: store
Device Capabilities and Performance Definitions JMF+JDF over https
Ganging Service
…
Other Optimization Services
Devices Database

***Figure 2:*** *Device Manufacturer submitting a Device to OaaS*

The device manufacturer can simply submit a JDF stating the various capabilities of the device, its unique ID, Family Name, Model Name and Version. This will be added to the set of available devices for every Optimizer present in the system at any given time. ***Excerpt 5*** illustrates an extract of a potential Scanner capabilities definition.

```
<Device DeviceID="HPSJN9120" ModelName="HP ScanJet N9120">
 <DeviceCap GenericAttributes="ID Class SettingsPolicy
    BestEffortExceptions OperatorInterventionExceptions
    MustHonorExceptions PartIDKeys DocIndex"
    Type="Scanning" CombinedMethod="None"
    ExecutionPolicy="AllFound">
  <Performance AverageAmount="600.0" Name="ExposedMedia" />
  <DevCaps Required="false" Context="Resource"
    Availability="Installed" Name="ScanParams"
    ResourceUpdate="None">
   <DevCap MinOccurs="1" Name="ScanParams" MaxOccurs="1">
    <EnumerationState ActionRefs="cd" MinOccurs="1"
      AllowedValueList="GrayScale CMYK RGB LAB"
      UserDisplay="Display" Editable="true"
      ID="colorspace" ListType="SingleValue" HasDefault="true"
      Name="OutputColorSpace"  MaxOccurs="1"/>
    <XYPairState MinOccurs="1" DefaultValue="600.0 600.0"
     AllowedValueList="100.0 100.0 300.0 300.0 600.0 600.0
                  1200.0 1200.0 2400.0 2400.0 4800.0 4800.0"
     UserDisplay="Display" Editable="true" ID="outputres"
     ListType="SingleValue" HasDefault="true"
     Name="OutputResolution" MaxOccurs="1"/>
   </DevCap>
   …
  </DevCaps>
 </DeviceCap>
</Device>
```

***Excerpt 5****: Hypothetical Scanner Capabilities Definition*

Now that we have examined the fundamental aspects of the platform we can delve into the description of some exemplar Optimization services that can be provided. The main objective is to illustrate how simply sending JMF and JDF content to OaaS is possible to perform several complex tasks without having to move assets and large files. This demonstrates that the future PSPs can leverage cloud services and have several pre-press stages executed in the cloud allowing them to access a wide variety of services and at the same time reducing costs allowing, for instance a pay-as-you-go strategy.

In the next sections we will introduce how Planning, Ganging and Batching Optimizers work just exchanging JDF data through JMF messages.

## Planning as a Service

Planning is the task where job requirements are transformed into production steps. Planning can be executed using several strategies:

1. it can be manually executed by a person in charge of print production;
2. it can be executed using a set of pre-built templates (e.g., with the aid of an MIS system);
3. it can be automatically negotiated by a software mapping requirements to machine capabilities.

***Figure 3*** illustrates a block diagram of the Planning operation.

Clearly, the latter is the most powerful but also most technically challenging. Since in our optimization approach we are trying to eliminate human interaction, we are considering the latter two cases.

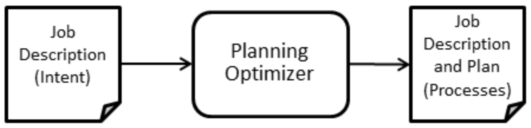


***Figure 3****: Planning Operation Block Diagram*

In OaaS the Planning Optimizer in addition to the Job Intent, it will be require having the list of Devices that are available in the system at that time or that are relevant to the planning. These can be passed as normal JDF resources with the exception of indicating them as implementation resources. The devices do not need any description, since their manufacturer, device model, version and ID can be used to fetch all the data from the database hosted by OaaS and updated by the device manufacturers. Once the Planning Optimizer has generated the plan(s), the output JDF will contain a set of **ProcessGroups** and it will remove the Device pointers since are now irrelevant.

Moreover, the Planning Optimizer can produce several plans and even present them with different level of preference. One way of doing this is to create **ProcessGroups** that have different Activation strategies. These are some cases that can be implemented:

- **Activation**="TestRunAndGo" this will be the preferred plan, since once verified it could automatically run.
- **Activation**="TestRun" this is a plan that needs to be tested but it will require human intervention to be executed. Having several of them can produce various validated alternatives to be chosen by an operator
- **Activation**="Informative" will force an operator to select one or multiple plans to be move in the testing and/or running phase.

## Ganging as a Service

Ganging is an operation applicable to two or more jobs. Jobs that are compatible, e.g. share the same substrate and color process, are jointly imposed on the same sheet or roll. This usually has several benefits that range from reduction of set-up costs (i.e., one set-up vs. many) and paper waste reduction (i.e., better usage of the sheet or web width).

When Ganging is offered as a service there are two main possibilities: first, the client selects jobs based on their characteristics and let the service to perform the generation of the ganged layout; second, the client sends as many jobs it can, irrespectively of their characteristics. In addition, the specific substrate format can be expressed or not, leaving to the ganging engine to figure out all the potential combinations, ***Figure 4*** illustrates the generic ganging operation.

As in the Planning Optimization we intend to leverage the JDF capabilities. In this case there is a specific process called **LayoutPreparation** that is supporting the Ganging Optimization. The idea is to have clients to send several jobs to the Ganging Optimizer and it will consider all the jobs that can be ganged and produce both a **LayoutPreparation** and **Layout** process for the combined job.

The **LayoutPreparation** takes into account all the job's specific finishing needs, as N-UP, binding and folding. The Layout instead

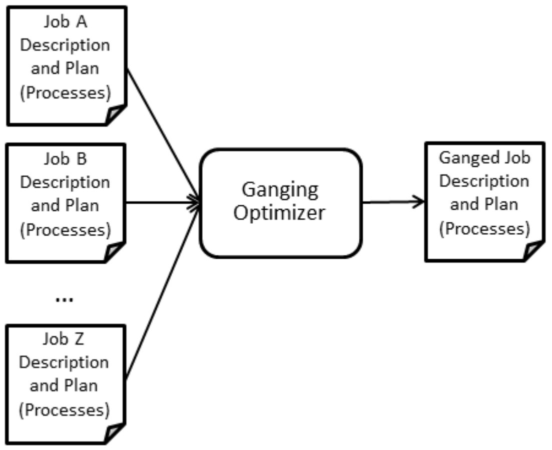

Job A Description and Plan (Processes)
Job B Description and Plan (Processes)
...
Job Z Description and Plan (Processes)
Ganging Optimizer
Ganged Job Description and Plan (Processes)

***Figure 4**: Ganging Operation Block Diagram*

represents the imposition signature. This encompasses several pages, in our case coming from several PDFs, positioned in automatically computed locations by the Optimizer. The proposed approach demonstrates that is not necessary to send large assets to the cloud services to perform the ganging, since the **LayoutPreparation** and **Layout** are only JDF. Hence, XML descriptions of the final layout using references to the PDF files but without loading their content. The JDF enabled imposition engine available at the PSP can then follow the instructions provided by the **Layout** process and generate the combined PDF.

## Batching as a Service

Batching is a process where several jobs are combined into a single one to reduce overhead in devices' setting and cleaning. Its main difference from Ganging is that the Jobs are not imposed into a single new job but are preserved as individual entities. In fact, batching can be performed with different levels of granularity:

- ***single device***; when batching is performed around a single device it is because that device is strategic in the workflow. Most of the time, this corresponds to the knowledge that this particular device represents the workflow's bottleneck and hence improving performance there translates into improving the overall performance.
- ***partial workflow or cell***; in this case optimization is done around a specific set of devices performing the similar or inter-correlated operations. Batching is particular common for situations where there are devices that perform cutting and folding, and having a consistent set of jobs allows to set these up once, saving a significant amount of time
- ***overall workflow***; this is clearly the most sophisticated case and tend to have an NP complete complexity. The optimizer in this case is more focused in joining jobs with similar characteristics rather than truly perform a global optimization.

The proposed service implementation for a Batching Optimizer could implement any of the above mentioned strategies since it will receive complete job descriptions containing all the Processes. As in the Planning Service, the client could add some Device Resources as implementation and indicate the service to focus around these specific processes. ***Figure 5*** illustrates a block diagram for the Batching operation.

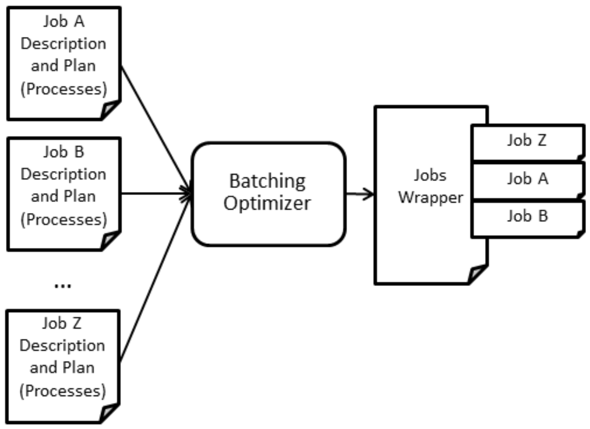

Job A Description and Plan (Processes)
Job B Description and Plan (Processes)
...
Job Z Description and Plan (Processes)
Batching Optimizer
Jobs Wrapper
Job Z
Job A
Job B

***Figure 5**: Batching* Operation Block Diagram

## Conclusions

The most important aspect of the proposed approach is the ability of moving pre-press optimization steps into the cloud without the need of moving large data elements, i.e. assets. This is particularly important since it currently represents a major barrier to service based solutions adoption from PSPs. Using OaaS and leveraging the JMF and JDF capabilities, as here presented, not only eliminates the need of sending/receiving large amount of data to/from the cloud, but also, removes many security concerns. Sensitive information contained in jobs, usually represented by the assets, is not exposed outside the PSP and the JDF/JMF data is exchanged using a secure HTTP connection.

The other benefit is represented by the ability of the cloud services to be updated and deployed by 3rd parties and made available to PSPs without the need of any notification or software download. PSPs can connect and request the list of available services and get back a message with the available optimizers.

## References

[1] Vidyanand Choudhary , "Software as a Service: Implications for Investment in Software Development," *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* , vol., no., pp.209a, Jan. 2007

[2] Gold, N.; Mohan, A.; Knight, C.; Munro, M.; , "Understanding service-oriented software," *Software, IEEE* , vol.21, no.2, pp. 71- 77, March-April 2004

[3] eXtensible Markup Language (XML), W3C, http://www.w3.org/TR/2008/REC-xml-20081126/

[4] Job Definition Format (JDF) and Job Messaging Format (JMF), CIP4, http://www.cip4.org/documents/jdf_specifications/index.html

## Author Biography

*Fabio Giannetti is a Senior Researcher at HP Labs. Since he joined HP in 2000, his focus has been on Digital Publishing, Variable Data Print and Printing Workflow Technologies. He has been actively involved in the research community releasing open-source tools as well as advancing the state of the art in numerous HP products. Fabio holds an MsEng (Hons) Computer Science from University of Genoa (Italy). Fabio represents HP at the W3C XSL Working Group.*