Integration-to-Job-Framework Migration Manual

Table of Contents

[Introduction 3](#_Toc471365831)

[Target Audience 3](#_Toc471365832)

[Disclaimer 4](#_Toc471365833)

[Prerequisites 4](#_Toc471365834)

[Summary 5](#_Toc471365835)

[Step-by-step Guide 5](#_Toc471365836)

[Migration Steps at a glance 5](#_Toc471365837)

[1. Set up Migration Cockpit in Business Manager 6](#_Toc471365838)

[1.1 Download and install the Cartridge 6](#_Toc471365839)

[1.2 Maintain permissions 6](#_Toc471365840)

[2. Migration of Workflow Components 7](#_Toc471365841)

[2.1 Component Migration Overview 7](#_Toc471365842)

[2.2 Steps to execute Workflow Component Migration 7](#_Toc471365843)

[3. Migration of Workflow Schedules 9](#_Toc471365844)

[3.1 Schedule Migration Overview 9](#_Toc471365845)

[3.2 Step by Step Migration 9](#_Toc471365846)

[4. Feature Comparison and Limitations 10](#_Toc471365847)

[4.1 Logging 10](#_Toc471365848)

[4.2 Workflow Schedule 10](#_Toc471365849)

[4.3 Workflow Plan 11](#_Toc471365850)

[4.4 Existing Implementations: Integration Framework environment emulation through Adapter class 11](#_Toc471365851)

[4.5 Automatic Migration vs. Re-Implementation 12](#_Toc471365852)

[5. Cleanup 13](#_Toc471365853)

[Job Framework - Good-to-Know 14](#_Toc471365854)

[Built-in Job Step Types 14](#_Toc471365855)

[Trigger and Monitor Jobs via Open Commerce API 15](#_Toc471365856)

[Working with Workflows 15](#_Toc471365857)

[Known Issues 16](#_Toc471365858)

[A. Appendix 17](#_Toc471365859)

[Example: steptypes.json 17](#_Toc471365860)

[Example: Business Manager representation of Step Type 18](#_Toc471365861)

# Introduction

It's been a while since the Community Suite (CS) **Integration Framework**, has seen the light of the world. The initial intent was to create generic-enough components which can be configured via a decent UI and combined to bigger, executable blocks (i.e. schedules), and add support for controlling, predictability, and flexibility. Since then, the CS Integration Framework has been well received and evolved over time. While it has been proven very useful per se, it also has some technical downsides, which led to the development of an out-of-the-box ***Salesforce Commerce Cloud*** platform pendant called **Job Framework**. This new framework has some features that don't exist in the ***Integration Framework***, while some features from the Integration Framework do not yet exist in the ***Job Framework***.

This document explains en-detail:

1. Prerequisites for the migration and corresponding considerations
2. Step-by-step migration of Workflow Components
3. Step-by-step-migration of Workflow Schedules
4. Features, that cannot be migrated from the Integration Framework

# Target Audience

* Developers, who will be concerned with migration of all Workflow Components using the provided tools
* Site Administrators, who will be concerned with migration of Workflow Schedules using the provided tools
* Business Users - Best Practices for Job Framework schedule setup and monitoring of execution (see also: [Documentation: Job management and scheduling](https://documentation.demandware.com/DOC3/topic/com.demandware.dochelp/Jobs/JobManagementandScheduling.html))

# Disclaimer

Similar to the Integration Framework itself, this migration guide and corresponding migration tools are not part of the official ***Salesforce Commerce Cloud*** platform, but custom code that has been written and is maintained by Community Suite members for your benefit, to help quickly migrate your existing background processes from the obsolete CS Integration Framework to the feature-richer and more future-proof Job Framework. As such, it is not subject of the official ***Salesforce Commerce Cloud*** support. In case problems occur, you may well report them in context of Bitbucket, but ***Salesforce Commerce Cloud*** has no obligation to fix them. Instead the community, which includes ***Salesforce Commerce Cloud*** employees as well as any other developer who wants to contribute their code, is asked to help out here.

It's is highly recommended to test migrated Job Step Types, aka Workflow Components, and schedules thoroughly, before you make them part of your daily procedures.

Switching from one framework to another does also come with an educational side, i.e. all personnel will need to be trained on the dos and don'ts, trouble shooting, etc. Please do not underestimate this aspect when planning and during your migration.

# Prerequisites

* Knowledge
  + Job Framework Documentation
  + Integration Framework Documentation
  + Basic understanding of the JSON Syntax
  + This Document
* Dependencies
  + An update of [bc\_library](https://bitbucket.org/demandware/demandware-library) might be required
  + The Migration supports all versions of the Integration Framework
* General Cleanup
  + Make migration as small as possible, only migrate necessary Components and Jobs
  + Delete unused Workflow Components (Code as well as CustomObjects)
  + Delete disabled (legacy) Job Schedules
* Planning (in case there are long-running jobs or if migration is taking longer or rather complex or if you pursue an integrative approach to mitigate risks)
  + Prioritization: A list that contains all Job Schedules, in the correct order for step-by-step conversion.
  + Definition of migration schedule:
    - Workflow Components
    - Tests
    - Final Job Schedule migration, also making sure that it doesn't fall into any maintenance windows of either Salesforce Commerce Cloud or involved 3rd party systems

# Summary

In short, the migration requires conversion of two kinds of entities:

1. **Workflow Components** will be converted into **Job Step Types**
   * Unlike Workflow Components, which are defined via Custom Objects of type **WorkflowComponentDefinition**, Job Step Types are defined through a ***steptypes.json*** file, found in every cartridge's root folder, that carries the corresponding implementation
   * Using the automated process and provided adapter scripts, the current Integration Framework implementation can be reused without being touched and even be ran in parallel via ***Job Framework*** and ***Integration Framework***
2. **Workflow Schedules** (and their component configurations) will be converted into **Job Schedules** (and their Job Steps)
   * Job Schedules are System Objects that are converted out of their Custom Object **WorkflowScheduleDefinition** counterparts.
   * Job Schedules either rely on standard, i.e. built-in components, or custom Job Step Type implementations.
   * Custom Job Step Type implementations have to be defined and implemented through a cartridge that is enabled in the cartridge path.

# Step-by-step Guide

## Migration Steps at a glance

1. Set up Migration Cockpit in Business Manager
2. Migrate Workflow Component Definitions (all Workflow Components must be migrated)
   * Distribute Job Step Type definitions to corresponding cartridges
   * Test and deploy the resulting code version (containing all Job Step Types)
3. Use the Migration Cockpit to make use of the semi-automated schedule migration
   * Import the resulting file via Business Manager. All Job Schedules will be imported as disabled.
   * Check for unsupported features
4. Create a prioritized list of the migrated schedules, the most frequent ones (e.g. Order Export) should get the highest priority
   * Handle each schedule in the resulting list as follows:
     + Disable Integration Framework Workflow Schedule
     + Run the job manually in Job Schedules, monitor execution and check results
     + Enable Job Schedule for given Job if successful
5. Cleanup

**Please be aware:** The steps outlined in this chapter depend on each other. Deviating from the stated will most likely lead to higher effort during migration, roll-out and configuration of the new Job Schedules.

## 1. Set up Migration Cockpit in Business Manager

### 1.1 Download and install the Cartridge

* The migration cartridge can be found in the [Integration Framework Community Suite Repository](https://bitbucket.org/demandware/integrationframework).
* Download or pull the code
* You will only need the **bm\_if\_migration\_cockpit** cartridge
  + no need to update or change other Integration Framework cartridges
* Migration Cockpit cartridge has a dependency towards **bm\_integrationframework**
* Deploy the cartridge to your target environment
* Add the cartridge to the Business Manager cartridge path

### 1.2 Maintain permissions

* Set corresponding permission for the Role that is supposed to do the migration: Go to Administration > Organization > Roles > [your role here] - Business Manager Modules, locate "CS Integration Framework Workflow Migration", make sure it is selected and click "Update". You need to do this in "Organization" context.
* **Note:** Don't forget to revoke permissions when migration is complete

## 2. Migration of Workflow Components

### 2.1 Component Migration Overview

Workflow Components can be seen as configurable building blocks for recurring jobs. They consist of their implementation (usually a Pipeline) and configuration parameters (accessed through the Pipeline dictionary). In order to achieve a decent UX, those parameters can be filled with values via Business Manager UI. But to be able show all configuration parameters (along with e.g. a short description or a drop-down field for possible values), metadata is needed that declares all parameters for generating a configuration UI in the Business Manager. Additionally, the Business Manager of course needs to know which Pipeline/Controller has to be run when the job is triggered.

This is what the **WorkflowComponentDefinition** Custom Object was responsible for in the Integration Framework. In the Job Framework, those components are now called **Job Step Types**.

In the Job Framework, the metadata contained in these objects will be delivered together with the cartridge that contains the implementation. This way the configuration is tied to the corresponding code more closely than through Custom Objects. Also, no metadata import is required when adding new Job Steps - adding the cartridge to the cartridge path will do.

In this step, we use the Migration Cockpit to transfer metadata from Custom Objects to the JSON notation: A file called "steptypes.json" will go in each cartridge that contains Job implementations and contain only their metadata. Since the Migration Cockpit cannot determine which cartridge contains which Job Step implementation, the user will have to split the resulting file and distribute the correct parts of it to the corresponding cartridges manually.

### 2.2 Steps to execute Workflow Component Migration

1. Open the Migration Cockpit. In Step "1. Export Workflow Components", click download button to generate the steptypes.json file.
2. View steptypes.json file in a Text Editor
   * The JSON contains one root object called step-types.
   * As there are only script-type Job Steps in this migration, the only step type used is script-module-step.
   * script-module-step defines an array containing all the custom code Job Steps.
3. Split the JSON file into cartridge-based configurations
   * The result of the migration script is one big *steptypes.json* file containing the step definitions for all your former workflow component definitions regardless of the cartridge the actual pipeline implementation is residing in. We strongly recommend to split the *steptypes.json* in case the pipelines living in separate cartridges and place a small *steptypes.json* file into each of the cartridges containing only the step definitions for which the pipeline lives in the same cartridge.
   * To do that, select the right script-module-step child objects from the resulting, big *steptypes.json* file for each cartridge that contains Job Steps
   * Move them out of the generated JSON file into a new one (using the same structure of step-types and script-module-step object)
   * Put the resulting JSON file in the root directory (not the "cartridge" folder) of the respective cartridge which holds the pipeline implementation for these steps
   * Example structure:
     1. /my\_cartridge\_a
        1. /cartridge
           1. /pipelines

JobPipelineA.xml

* + - 1. steptypes.json (containing only step definitions for job implementation in JobPipelineA.xml)
    1. /my\_cartridge\_b
       1. /cartridge
          1. /pipelines

JobPipelineB.xml

* + - 1. steptypes.json (containing only step definitions for job implementation in JobPipelineB.xml)

1. Deploy all affected cartridges and activate new Code Version if necessary
2. Review Results
   * In Business Manager, go to Administration -> Job Schedules (Beta)
   * Create new Job (call it "Migration Test" or similar), you will face the Job Editor now
   * Click on "Configure a Step", you will see the list of all Job Step Types
   * Go through the list of the migrated Job Step Types, select them one by one, and check if all parameters are listed in the form
   * Test Job Configuration can be deleted again after all Job Steps are verified
3. Job Step Types are now migrated to the Job Framework.

**Important:** Do not change any of the resulting Job IDs (@type-id) attributes at this point, as the Job Step IDs will be used by the migrated Schedules later (Step 3 of this migration).

**Example step type split:** Given you have three Job Pipelines (Export-Oders, Import-OrderStatusUpdate, Import-Inventory) distributed by two Cartridges (bc\_export, bc\_import), two resulting steptypes.json files will have to be created out of the exported JSON file:

* + One will go to the root of bc\_export and contain the definition for Export-Orders,
  + the other one will go to bc\_import and contain the definition for Import-OrderStatusUpdate and Import-Inventory.

**Example step type result:** Please see Appendix A.

## 

## 3. Migration of Workflow Schedules

### 3.1 Schedule Migration Overview

A Workflow Schedule consists of two main parts:

1. The Configuration and Schedule, containing information about when and how often the job is run, what Sites it should get as context, how to log, react on and notify about unexpected behaviour.
2. The Workflow, defining an order of steps (Workflow Components) that are executed one after the other. Those Workflows again contain configuration, filling the parameters that are defined for each component with values.

Our goal in this step is to convert those Workflow Schedules to Job Steps in the new Job Framework, resulting in an as-complete-as-possible representation in the new environment. Unfortunately, in some cases a perfect 1:1 conversion won't be possible, but using the Migration Cockpit can help to get very close to that. (See Chapter 4 for a detailed feature comparison.)

Fortunately, it is possible to operate both Integration Framework and Job Framework next to each other. While users must not enable Jobs in both environments at the same time, it is possible to disable a certain Job in one environment and enable it in the other. This way it is possible to do a step-by-step migration of each Job, one after the other. Additionally, if it comes to issues with the Job Framework, the Job can be run via Integration Framework just like before, ensuring there is no downtime until jobs are migrated properly.

While this tutorial describes automatic migration, please consider manual migration (by re-implementing or refactoring) jobs that e.g. face requirement changes in the near future or just need to be refactored "some day". Migrating to Job Framework is a great opportunity to do some cleanup - and learn how to use the new Framework at the same time, as there are several Script API differences coming along with the new Framework.

In order to be able to run Integration Framework Jobs without any code changes, automatic migration will make use of an Adapter class emulating the Integration Framework environment. (See Chapter 4 for details.)

### 3.2 Step by Step Migration

1. Navigate to Migration Cockpit.
2. In Step "3. Export Workflow Schedules", click download button to generate the job-definitions.xml file.
3. View XML file in a Text Editor, check for comments
   * Comments indicate that there was a problem during the conversion. See section 4.2 for details.
   * In order to fix the issue, reconfiguration and a new Export might be required.
4. With all issues fixed, head to Administration -> Operations >  Import & Export.
5. Using the Import & Export tool, upload the XML file you received from the Migration Cockpit.
6. In the "Job Schedules" section, Import the uploaded XML file.
7. Navigate to Administration -> Operations -> Job Schedules
8. In the desired order, execute following steps for each of the generated Jobs:
   * Open the Job in the Job Schedules tool, open "Step Configurator" tab
   * There must not appear any steps that are marked red or having an exclamation mark next to them.  
     If this is the case, head to the corresponding Integration Framework Workflow Schedule and disable it, to make sure both workflows won't run in parallel.
   * Ensure the Workflow is currently not running (check the Workflow Plan)
   * Navigate to the imported Job Schedule again and run it
   * Monitor the execution using Administration -> Operations -> Job History
   * If successful, activate the Job schedule for regular execution.
9. The migration is done then all Workflow Schedules are disabled and all corresponding Job Schedules are enabled and running properly.

## 4. Feature Comparison and Limitations

### 4.1 Logging

* While the CS Integration Framework had different Log levels, Job Schedules only have one
* Log entries will be prefixed with the corresponding Log level, but filtering/suppressing log entries by severity is not possible out of the box anymore.
* Migration Cockpit emulates a Verbose Mode that disables Log levels below WARNING in order to reduce excessive logging
  + Please monitor logfiles and edit your log behaviour appropriately if necessary
* Also, defining own log files is not supported (and not recommended) any longer.
  + The Job Framework creates one log file for each run
  + The log file can easily be accessed through the business manager

### 4.2 Workflow Schedule

#### 4.2.1 SUSPEND exit status

This status is not supported anymore. Jobs using this mechanism will have to be reimplemented and cannot be migrated automatically. Since this exit status is defined via Demandware Script as a return value, the Migration Cockpit can not react on this kind of limitation. Job step types and Job steps will be exported normally, but the job behaviour will change. Be sure to check your code for usage of SUSPEND exit status if you run into issues.

Additionally, there are three Integration Framework standard components (DateCondition, TimeCondition, DateTimeCondition) that use this exit status. These components will not be supported by the Job Framework, and Workflow Schedules using one of them will be marked with a comment in the exported XML.

#### 4.2.2 Disabled Job Steps

In the Integration Framework, each Component of a Workflow Schedule could be disabled separately. This is not possible in the Job Framework, and therefore disabled Workflow Components (Job Steps) will not be migrated. Whenever the Migration Cockpit export faces a disabled Job Step, it will be exported as a section in comments (still containing the definition for each step).   
Please search for commented sections in the job-definitions.xml file! If you want to migrate those steps, feel free to remove the comments. However, it is recommended to enable/disable corresponding steps before you run the export with this limitation in mind.  
If you need the possibility to enable/disable certain Job Steps, please introduce a parameter that tells the step whether it should execute or not, and implement conditional execution for that step.

### 4.3 Workflow Plan

With the Job Framework, It is not possible to generate a plan for future executions of Jobs. So instead of a plan, there is the Job History only, showing executions from the present or past.

### 4.4 Existing Implementations: Integration Framework environment emulation through Adapter class

The CS Integration Framework, as an actual framework, of course defined its own interfaces and objects. So all jobs that were implemented against the Framework expect certain functionality, or in other words have certain dependencies. In order to ensure that implementation does not have to be touched, those dependencies have to be emulated after migrating to the Job Framework.

The main focus lies on the CurrentWorkflowComponentInstance object, which is passed through the pipeline dictionary. This object is available in the Job Framework, provided by an adapter class. Preparing/Mocking the Object happens through Adapter class, which is used as an entry point for every migrated Job Step. Besides other custom job parameters, the target Job Pipeline/Controller is passed through an additional Step Parameter (Action). The configured pipeline will be called after mocking the object. This approach ensures that no code changes are necessary for the migration. As the adapter class (along with all standard components provided by the Integration Framework) is part of the Integration Framework cartridge - which has to stay in the cartridge path.

### 4.5 Automatic Migration vs. Re-Implementation

While the migration takes advantage from a lot benefits provided by the new Job Framework, there are some drawbacks that can only be solved by re-implementation of Job Steps and manual re-configuration of Job Schedules.

**Job Steps**

* The maximum timeout for Script Pipelets within Pipelines is limited to 60 min (3,600 sec). Some legacy components may have workarounds implemented to deal with that problem and by doing that, may have accidentally or acceptingly put extra load on servers, which in turn could cause performance problems. Please consider re-implementation.
* Unlike with Workflow Components, you can define maximum run times for Job Step Types. That helps to limit the use of job threads to an acceptable level.
* Code readability/maintainability may have suffered from the requirement to use Pipelines.
* To be able to share data between Steps a Map data structure is available in the pipeline dictionary under the key "JobContext". That said, you may want to come up with other ways than the file system or custom objects, to pass results from one Step A to Step B. Please use this feature it with caution though.
* The SUSPEND exit code for Job Steps is not supported at all. If your Workflow Component has been using it, you need to come up with an alternative implementation approach.  
     
  See <https://documentation.demandware.com/DOC2/topic/com.demandware.dochelp/Jobs/CreatingCustomJobSteps.html> for detailed description on how to write customer job steps.

**Job Schedules**

* Sometimes workflows on different levels, i.e. site vs. organization, different sites, etc., had to be chained. Typically it has been achieved by feeble timing between independent jobs. With the new framework it can be configured properly.
* There was only one email distribution list for all status notifications. With the new framework, you can assign different lists to different statuses.
* There platform Job Step Types that can be used to streamline processes. E.g. once you got some confidence in your product data replication processes from your back end to staging, or you run jobs to help preparing the data for production every day, you can now easily combine it with data replication directly.
* With the old framework it wasn't quite possible to set appropriate resource locks, which mark resources as exclusively locked by the current job. That sometimes led to inconsistent data. The only way to prevent that from happening was it to set the resource lock for all platform Workflow Schedules (the once that triggered the Integration Framework Jobs); a rather bad and not recommendable practice. With the new framework, you can set the resource lock per schedule, which is way more flexible and safe. E.g. when you import catalog data, you want to set a resource lock on catalogs so no data replication could be started while the catalog import is still in progress.  
    
  See <https://documentation.demandware.com/DOC2/topic/com.demandware.dochelp/Jobs/CreatingaNewJobSchedule.html> for detailed description on how to setup new job schedules.

## 5. Cleanup

* Before executing this step, be sure that ALL jobs and steps are up and running correctly using Job Framework
* Make a Site and Metadata Backup (especially regarding CustomObjects)
* Delete all Integration Framework related Custom Objects (WorkflowComponent\*, WorkflowSchedule\*)
* Disable Business Manager Tools (by revoking access rights):
  + Workflow Schedules
  + Workflow Plan
* Do not remove/delete Integration Framework Cartridges!
  + See 4.4 for details on Adapter Class
  + Migration Cockpit Cartridge can be removed though
  + Only if there is no more job depending on the adapter, those cartridges can be removed.

# Job Framework - Good-to-Know

## Built-in Job Step Types

Besides your migrated or re-implemented Job Step Types, you will find some built-in Job Step Types, that will help with you operational tasks.

|  |  |  |
| --- | --- | --- |
| **Name/ID** | **Description** | **Remarks** |
| ExecutePipeline | Executes a pipeline. The name and start node of the pipeline has to be configured at parameter 'ExecutePipeline.Pipeline'. | This Job Step can be used to migrate the current platform Job Schedules to the Job Framework based ones. In this case the Job Schedule will contain only one flow and step using the parameters that have been added to the old ones. |
| ExecutePreconfiguredCodeReplicationProcess |  | Let you trigger an already created Code Replication Process, identified by its ID, that has been configured with Activation Type 'Job Step'. |
| ExecutePreconfiguredDataReplicationProcess |  | Let you trigger an already created Data Replication Process, identified by its ID, that has been configured with Activation Type 'Job Step'. |
| ExecuteScriptModule | Executes a function exported by a script module. The module ID has to be configured at parameter 'ExecuteScriptModule.Module'. | Recommended to be used to execute Script modules that don't have a list of parameters defined via ***jobsteps.json***. |
| IncludeStepsFromJob | Includes steps from another job. | Can be used to "run" a Job, identified by its ID in context of this step. E.g. it can be used to avoid redudant  configurations. |
| UndoPreconfiguredCodeReplicationProcess |  | Roll back of pre-configured code replication processes |
| UndoPreconfiguredDataReplicationProcess |  | Roll back of pre-configured data replication processes |

**Please note:** The above is only the initial set of Platform Job Step Types and going forward you may see additional ones appearing. Refer to the platform documentation to stay current in this regard: <https://documentation.demandware.com/DOC2/topic/com.demandware.dochelp/Jobs/StepDefinition.html>

## Trigger and Monitor Jobs via Open Commerce API

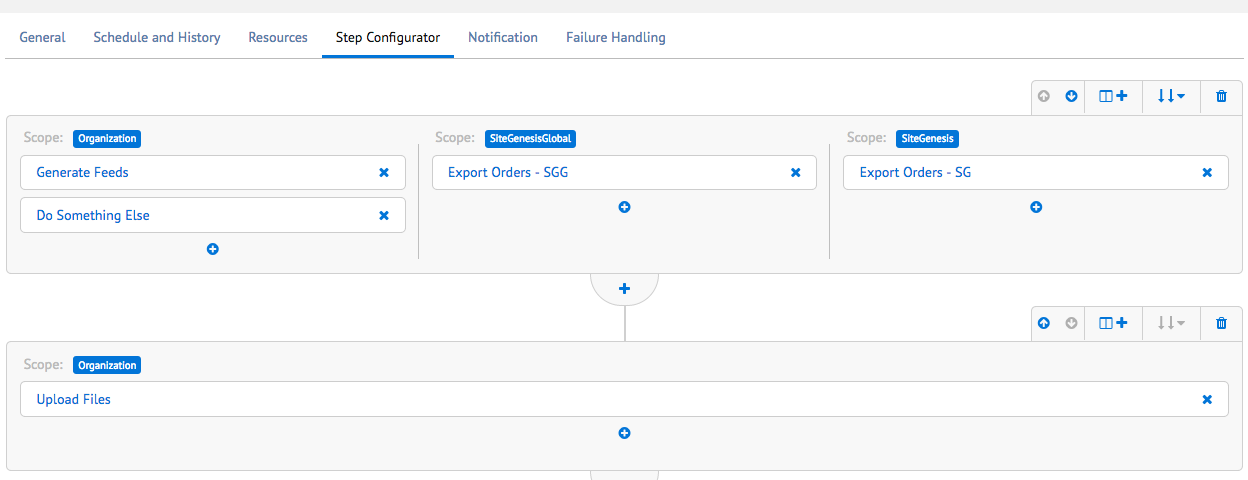
With the Job Framework, external parties can now trigger Job Executions via OCAPI (<https://documentation.demandware.com/DOC2/topic/com.demandware.dochelp/OCAPI/17.1/data/Resources/Jobs.html>). I.e. just like any other permitted back end system, a Product Information Management System can now push out new product data and immediately trigger the job inside SFCC that would consume the data and replicate everything to production. Please make sure that only a very close and trusted group of external systems get this type of access, as there is no fine grained permission control on those jobs, i.e. the one that got the permission to execute a Job, can basically execute all jobs if it knows it by ID.

Using the Job Resource, it is also possible to monitor the execution of these jobs. That helps to make external systems aware of completion of jobs and react on any given exit status. Using this API, it would also be possible to create dashboards showing the current status of jobs, errors, duration, etc.

## Working with Workflows

First off, some rather technical definitions that are important to understand:

* A job can consist of 1 - many workflows
* Each Workflow can have a different scope, i.e. Site vs.Organization, different Sites, etc.
* Workflows can be parallelized
* Job Steps within Workflows are executed sequentially
* Different Workflow Boxes, are executed sequentially



The above image depicts a Job that runs three Workflows in parallel before the last Workflow, the one at the bottom, gets executed.

**Please note:** Running Workflows on the same level in parallel is the the default. In case you need them to run sequentially, e.g. because both streams would cannibalize their resources, you explicitly need to configure it this way. Just use the toggle box (second from the right on top of workflow boxes) to adjust the setting accordingly.

The above gives you a very powerful framework at hand to setup and fine-tune your jobs. The prerequisite for using it at full extent is to have a thorough understanding of dependencies between different Job Steps and Flows.

## Known Issues

* Support for parameter data type password. A corresponding ticket (APP-40717) has been filed with ***Salesforce Commerce Cloud*** and we are hopeful to see it in one of the next platform releases. Until that happens, all password parameters have to be treated like string and the values are just as visible in the forms as strings. In case you find it unacceptable, you may well wait with your migration until password data types are supported.
* A pendant to the Integration Framework Workflow Plan Panel is missing. I.e. it is hard to see for what time the jobs have been scheduled. This type of panel will most likely be release in the course of 2017.

# A. Appendix

## Example: steptypes.json

Shows configuration for a cartridge that contains one Job Step (StandardComponents-CleanUpFiles) with 5 parameters. Would go to bc\_myCartridge\steptypes.json.

**{**

**"step-types": {**

**"script-module-step": [{**

**"@type-id": "custom.CleanUpFiles",**

**"module": "bc\_integrationframework/cartridge/scripts/workflow/legacy/PipelineStepRunner",**

**"function": "execute",**

**"parameters": {**

**"parameters": [{**

**"@name": "Action",**

**"@description": "Legacy action of the community suite IF, to be called through the new Step Execution",**

**"@type": "string",**

**"@required": "true",**

**"enum-values": {**

**"value": [**

**"StandardComponents-CleanUpFiles"**

**]**

**}**

**}, {**

**"@name": "FilePattern",**

**"@description": "File name pattern (Default is \".\*\")",**

**"@type": "string",**

**"@required": false,**

**"@trim": true**

**}, {**

**"@name": "DirectoriesToPurge",**

**"@description": "Directories to purge",**

**"@type": "string",**

**"@required": false,**

**"@trim": true**

**}, {**

**"@name": "DirectoriesToArchive",**

**"@description": "Directories to archive",**

**"@type": "string",**

**"@required": false,**

**"@trim": true**

**}, {**

**"@name": "DaysToKeep",**

**"@description": "Number of days to keep old files",**

**"@type": "double",**

**"@required": true,**

**"@trim": true**

**}]**

**},**

**"status-codes": {**

**"status": [{**

**"@code": "ERROR",**

**"description": "Used when an error occurred."**

**}, {**

**"@code": "OK",**

**"description": "Used when everything went well."**

**}, {**

**"@code": "WARN",**

**"description": "Used when small, but acceptable problems occurred."**

**}, {**

**"@code": "SUSPEND",**

**"description": "Used when nothing unexpected happened but subsequent steps should be bypassed."**

**}]**

**}**

**}]**

**}**

}

## Example: Business Manager representation of Step Type

Shows configuration for JSON (ID, Description and Action being standard attributes).

