**Terminology**

It's not always fun and exciting, but sometimes you just have to learn a few terms and concepts up front, then solidify them as the learning process continues.

* **Orchestration Plan Definition** - A set of tasks grouped logically together in a plan, used to fulfill orders across one or more fulfillment systems. Note that the tasks processed are for the fulfillment requests that were created by the decomposition process. (That is, decomposition feeds into the orchestration process.)
* **Orchestration Item Definition** - Tasks that make up an orchestration plan. They are executed in sequence (synchronous) or in parallel (asynchronous). There are 5 types: Milestone, Auto-Task, Callout, Manual Task, and Push Event.
* **Orchestration Scenario** - Used to determine when an orchestration plan should execute. They are tied to a product and an action (such as Add, Modify, Disconnect, or NoChange). For example, execute ScheduleRouterInstall (plan) if HSRouter is added (action) to an order. Orchestration Scenarios optionally support sub-actions too.

**Types of Orchestration Item Definitions**

**Milestone -** A simple task used as a marker in the fulfilment process.

There is a saying "Love never fails"... well Milestones don't either! They always evaluate to True.

**Manual Task** - As implied by the name, a Manual Task is used when user input is needed at a particular place in the fulfilment flow. Common examples are supervisor approval, a credit check, or purchasing more coffee for the break room. Manual Tasks are placed in Manual Queues, where they can be taken off the queue, worked on, and eventually completed.

**Auto-Task** - Auto-tasks are tasks that automatically execute by invoking an Apex class associated with the orchestration item. The most common example for an Auto-Task is creating assets for an order as one of the last steps in order fulfillment. Industries Order Management provides an Apex class called **XOMAutoTaskAssetizer** out of the box to accomplish this.

It's considered best practice to create assets as the second to last task in the orchestration plan. (Basically, to assetize just prior to completing orchestration with a Milestone that can't fail.)

As you may have surmised by now, auto-tasks have nothing to do with automobile-related tasks on an assembly line.

**Callout** - A Callout is an automated interaction with an external fulfilment system using a web service or RESTful API. Again, common examples include billing systems, scheduling systems, inventory systems, etc. Although they are often third party systems, they could be internal systems as well.

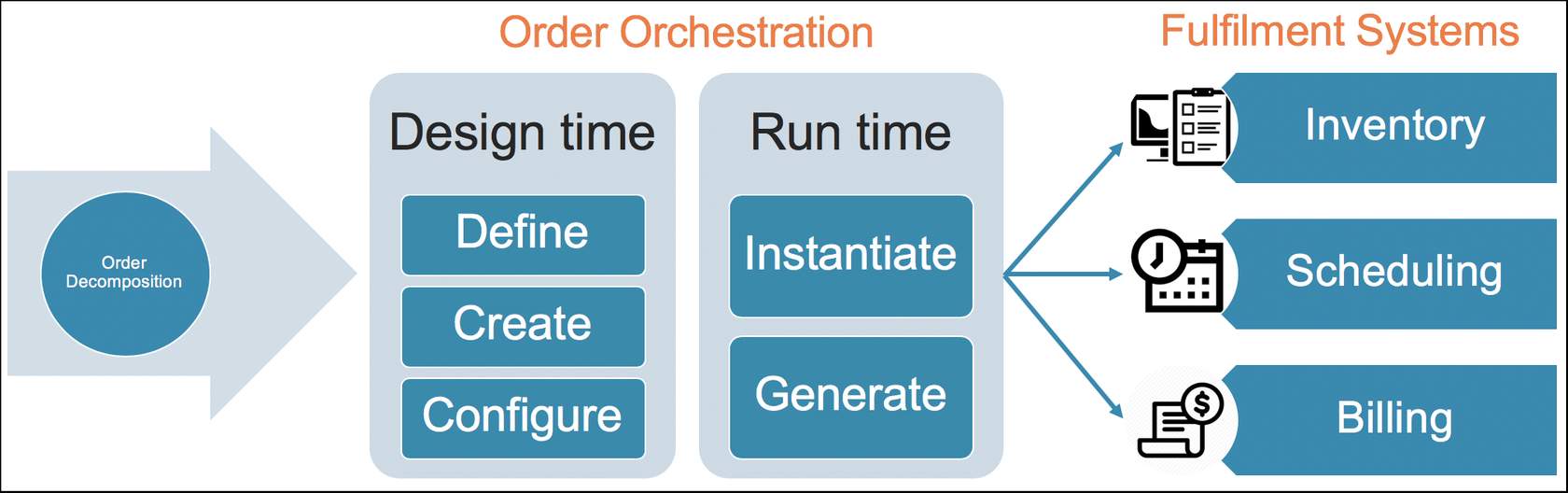
**Push Event** - A push event is activated when an external system or a Salesforce user makes a specific change to the order associated with a push event orchestration item. A push event pauses until an event (e.g. event condition) evaluates to true. Triggers are used to constantly evaluate the event condition.

**High-Level View**

Before going further, let's take another breath, and cover order orchestration from two different perspectives:

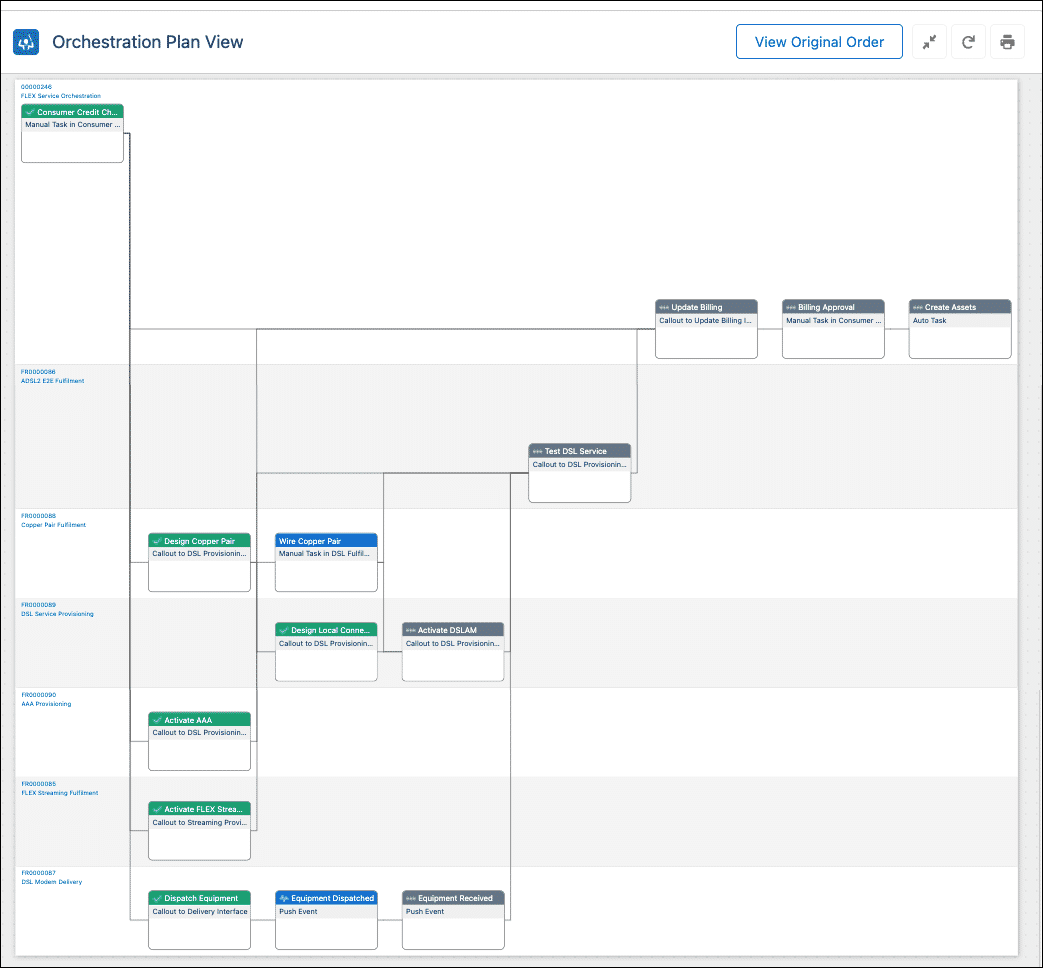
1. Process view
2. Plan view

**Process View**  
Logically, the diagram below flows from left to right.

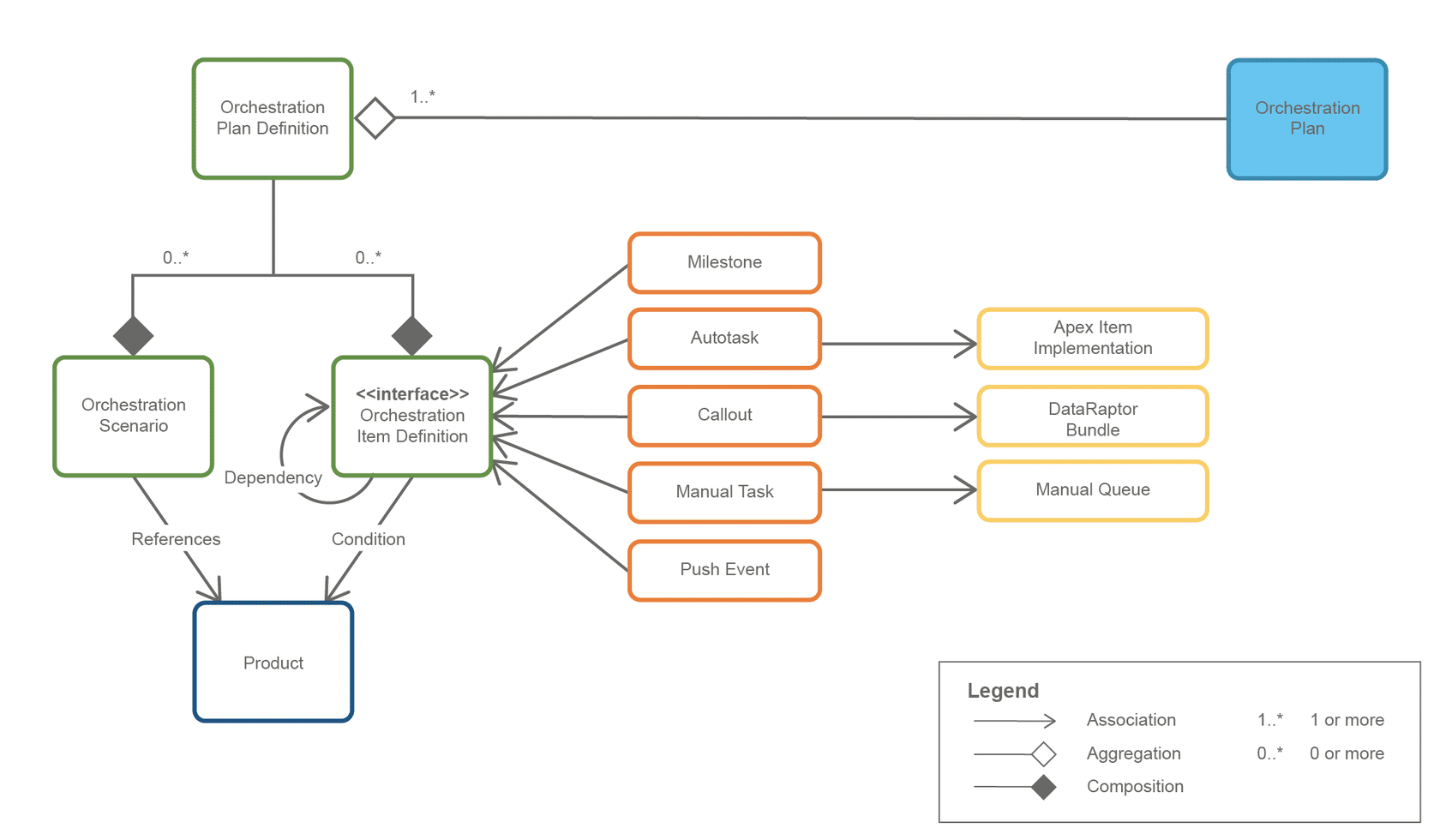


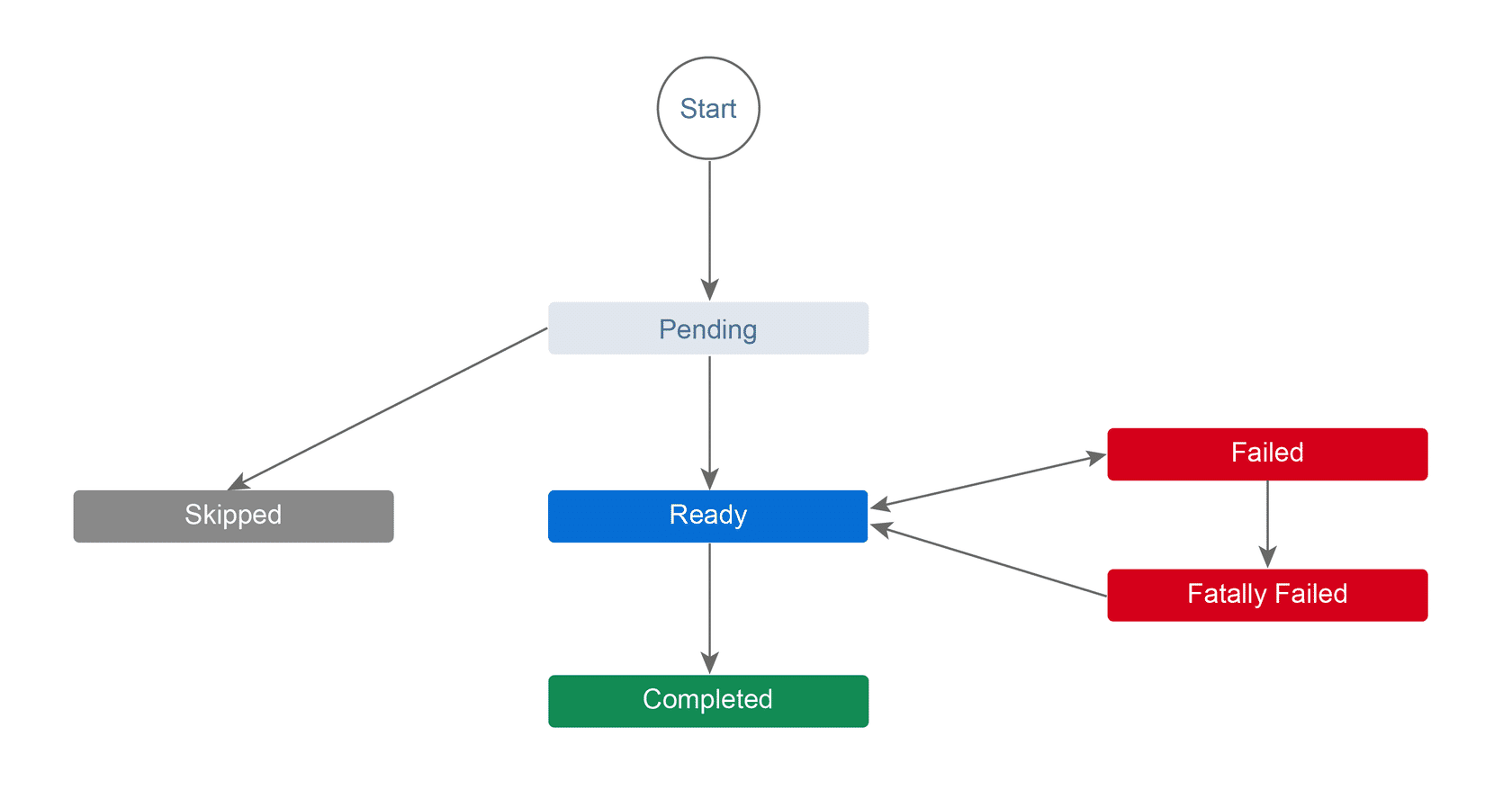
In most cases, run-time orchestration objects have a corresponding design-time definition object. The run-time object is said to be an “instance of” the definition object.

**Plan View**  
This view of an Orchestration Plan puts much of what has been covered thus far into a single diagram.



* Notice the following important points from the example plan shown above.
* **Color-coded status**- Each task is represented by a color-coded card in the plan view. The colors of the card depict the current status.
  + **Green** - Completed
  + **Blue** - Ready
  + **Light Gray** - Pending
  + **Dark Gray** - Skipped (not shown)
  + **Red** - Failed (not shown)
* **Swimlanes** - Horizontal grouping of items in the plan. Each Orchestration Plan Definition is depicted by a swim lane, and typically represents a single downstream system for order fulfillment. (Billing, Activation, etc.) Often, there is an end-to-end plan that organizes or invokes other orchestration plan definitions, which creates multiple swim lanes.
* **Types of Tasks** - Notice below each task name in the card header is the type of task. The example plan includes Milestones, Manual Tasks, Auto Tasks, and Callouts. (A Push Event is not in the example plan, but is part of a hands-on exercise.)
* **Dependencies** - Lines connecting cards show dependencies, such as tasks that depend on the completion of other tasks before they can begin. When you hover over a task in the Orchestration Plan view, the dependencies for that task are highlighted in red.





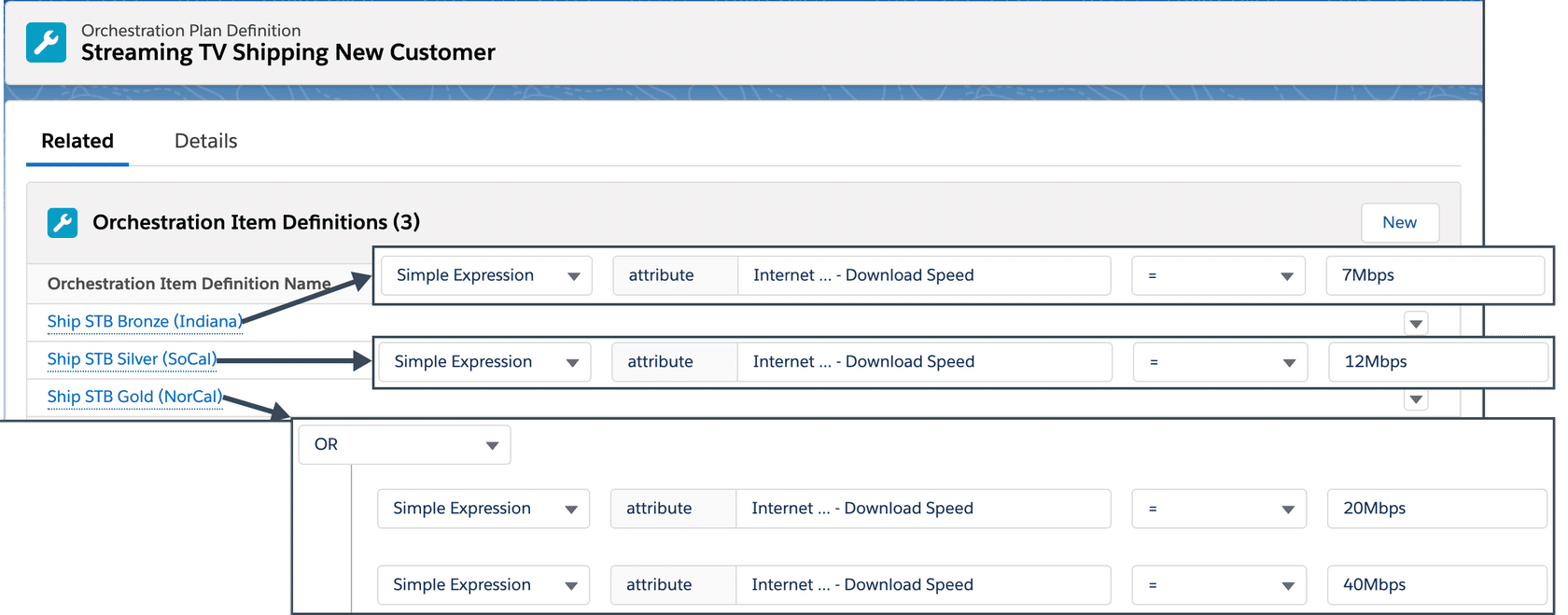
**Conditions**

Orchestration provides two types of conditions to assist with the control of order fulfilment.

1. **Orchestration Item Definitions**
2. **Orchestration Scenarios**

The features and mechanics of establishing conditions are identical to conditions for order decomposition.

**Orchestration Item Definition Conditions**  
Orchestration Item Definition conditions control whether the task should execute if the condition is met, or skipped if the condition is not met.



**Orchestration Scenario Conditions**  
Thus far, you have seen orchestration scenarios used to determine when an orchestration plan should execute based solely on a product, and action (or actions). However, you can also specify conditions. One or more conditions can be specified that examine the product attribute or field against the values you specify. This provides finer grain control over the scenarios and thus the presence of the orchestration plan swimlanes on the generated orchestration plan.

**Manual Tasks**  
Manual Tasks require some level of human intervention. Although that can be as simple as supervisor approval, it can also include a custom execution URL that runs a custom OmniScript. For example, one that steps through credit approval or warehouse inventory checks. So, manual tasks don't necessarily mean you're using a low-tech approach. Instead, Manual Tasks can be cutting-edge with a high-tech OmniScript!!

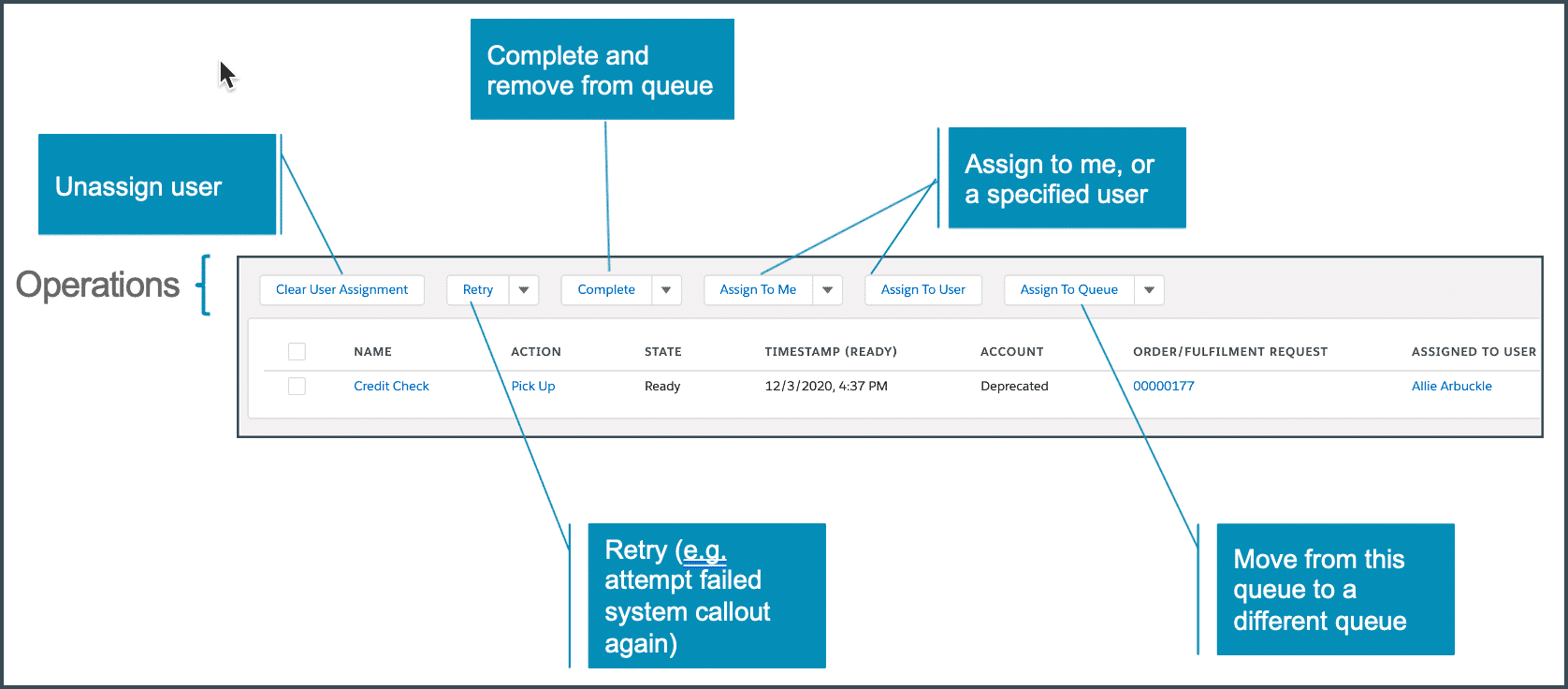
**Manual Queues**  
Manual Tasks get placed in Manual Queues. Unlike Orchestration Queues, which are created for you as part of the managed package installation, Manual Queues are created by Administrators or Fulfillment Designers. Creating Manual Queues is extremely easy to do. It is common practice to create several queues to help balance workloads. For example, you can create a queue for each swim lane. That way, a resource that is heavily used (such as inventory) does not impose bandwidth limitations for one that is not (scheduling home installations).

**Adding Manual Tasks to Manual Queues**  
In most cases, adding tasks to queues is easy to do. From the **Related** tab on the correct **Orchestration Plan Definition**, click **New** to create a Manual Task.

There are a few other things to note before moving on.

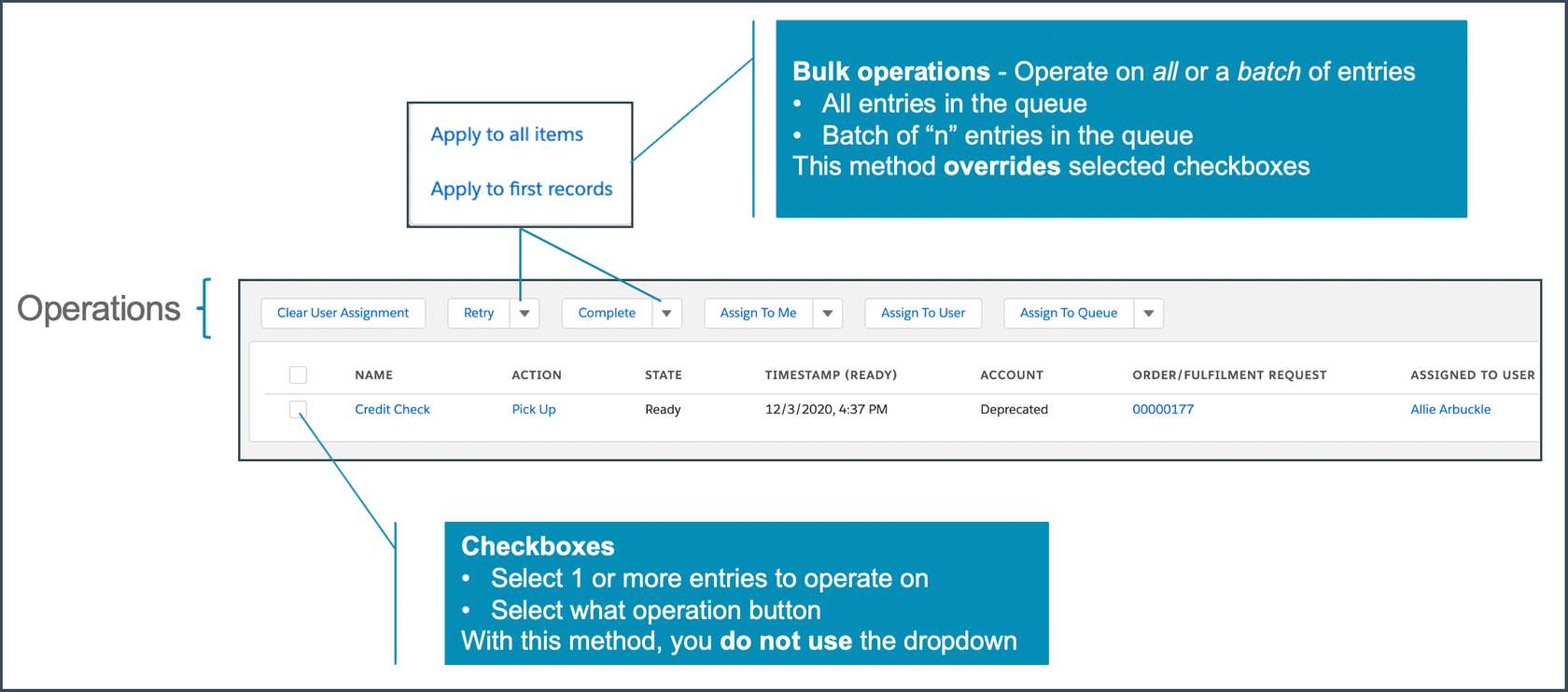
* **Point of No Return** (PONR) - If PONR is checked, once the task is completed, the status of the associated order (or order-item) is marked "PONR". PONR is key when using features such as canceling or amending in-flight orders. Once reached, IOM communicates to CPQ that the order (or order-item) cannot be canceled. (Note that other line items may be canceled, depending on the configuration. That is, part of an order may still be can be canceled. Order cancellation is covered later.)
* **Custom Task Execution URL** - URL to execute when the task is picked up from the queue. The URL points to a VisualForce page or OmniScript.
* **Rollback** fields are discussed later in the Order Cancellation section. In-flight amendments function very similar to cancellations (but are not covered in the Industries Order Management curriculum.)

**Manual Queues - Operations**  
Once Manual Tasks are placed in Manual Queues, there are several operations (or actions) you can perform against the tasks.



**Manual Queues - Specification**  
Before clicking any buttons to perform various actions against items in the queue, you must specify which items. After all, there could be dozens, hundreds, or even thousands of items in a queue! There are two ways to specify what items in the queue to work with:

1. **Checkboxes** - Select one or more items, then click an action button.
2. **Bulk Operations** - Operate on all or a batch of "n" entries in the queue.



|  |  |
| --- | --- |
| ⚠ | Bulk operations override selected checkboxes! For example, if there are 100 tasks in the queue, and you select 10 of them, and then the bulk "**Apply to all items"** from the Complete drop-down, all 100 tasks are marked as Complete (not just the 10 you had previously selected). |

|  |  |
| --- | --- |
| 💡 | *Tip!* Consider using Manual Tasks (or Auto Tasks that are configured to intentionally fail) as a stopping point for testing orchestration plans during development. Similar to setting breakpoints in code when debugging, you can force your orchestration plan to pause. Is the plan progressing as intended? Or are more modifications needed prior to continuing beyond the Manual Task? Note that the Custom Task Execution URL is *not* a mandatory field. So if using a Manual Task for this purpose, you can leave the custom URL field blank. There is no need to launch any code. |

Unlike Manual Tasks that wait on user intervention, Auto Tasks automatically execute by invoking an Apex class associated with the orchestration item. They are used for the internal processing of data.

**Odd Words**

* Gubbins - Object with little to no value.
* Snollygoster - Clever, unscrupulous person.

Salesforce Industries saw fit to create another odd word: **Assetize** (or assetizer). It may be an odd word used in the communications industry, but it's important, and very common when it comes to Auto Tasks.

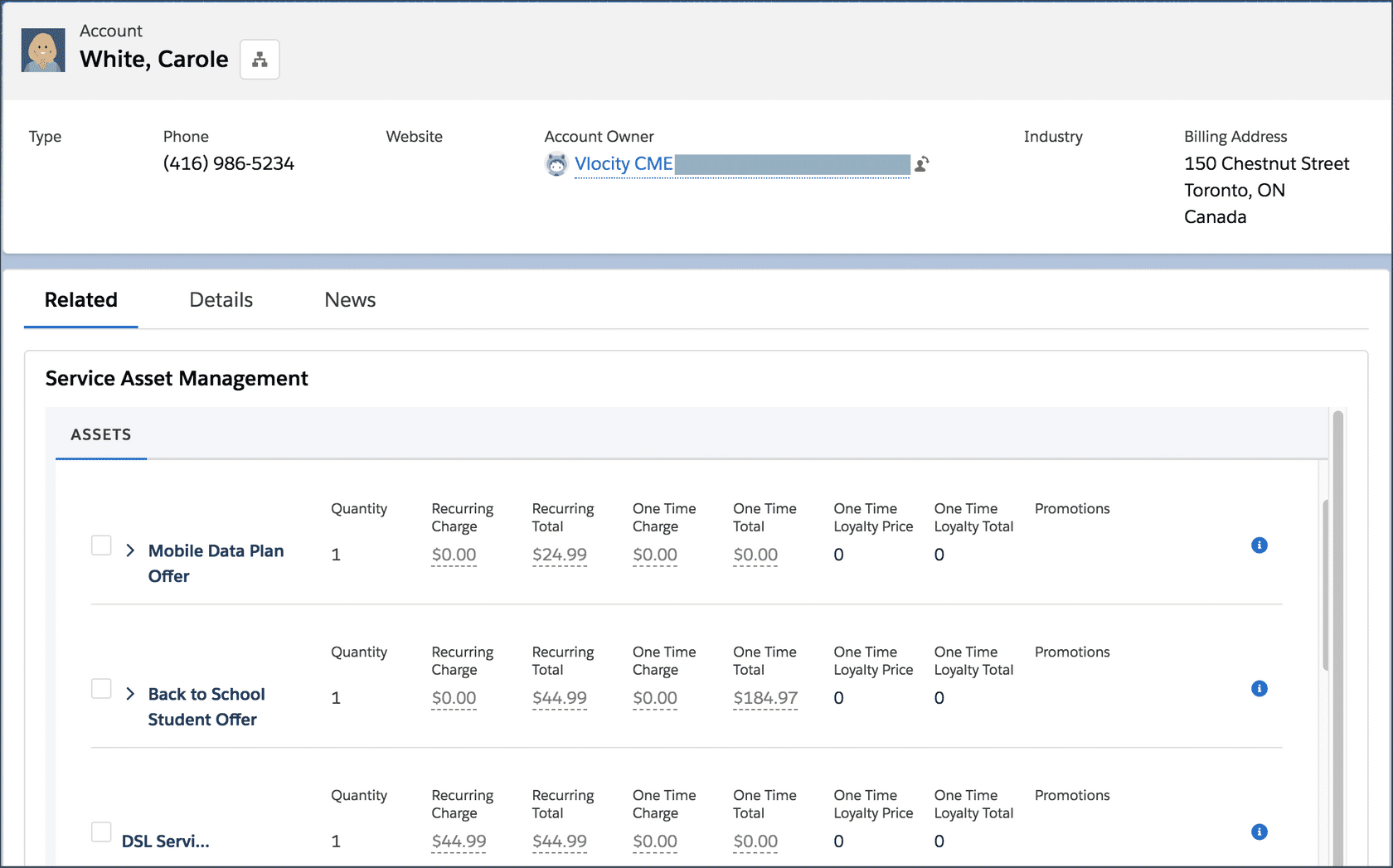
* **Assets** - A specific model or type of product that a customer owns
* **Assetization** - The process by which Industries Order Management converts OrderItems into Assets and Fulfillment Request Lines (FRLs) into Inventory Items

**XOMAutoTaskAssetizer**  
A common use of the Auto Task is to create assets as one of the last steps of order processing.

Industries Order Management provides an out-of-the-box item implementation called **Assetize** and an Apex class called **XOMAutoTaskAssetizer** to perform this function. XOMAutoTaskAssetizer creates assets from the corresponding order at the end of processing. In fact, it's considered best practice to assetize as the second to last step in the orchestration plan. The last task is typically "Complete Order", a Milestone that cannot fail.

|  |  |
| --- | --- |
| 📝 | The Assetize item implementation only assetizes commercial products. Technical products are not assetized. |

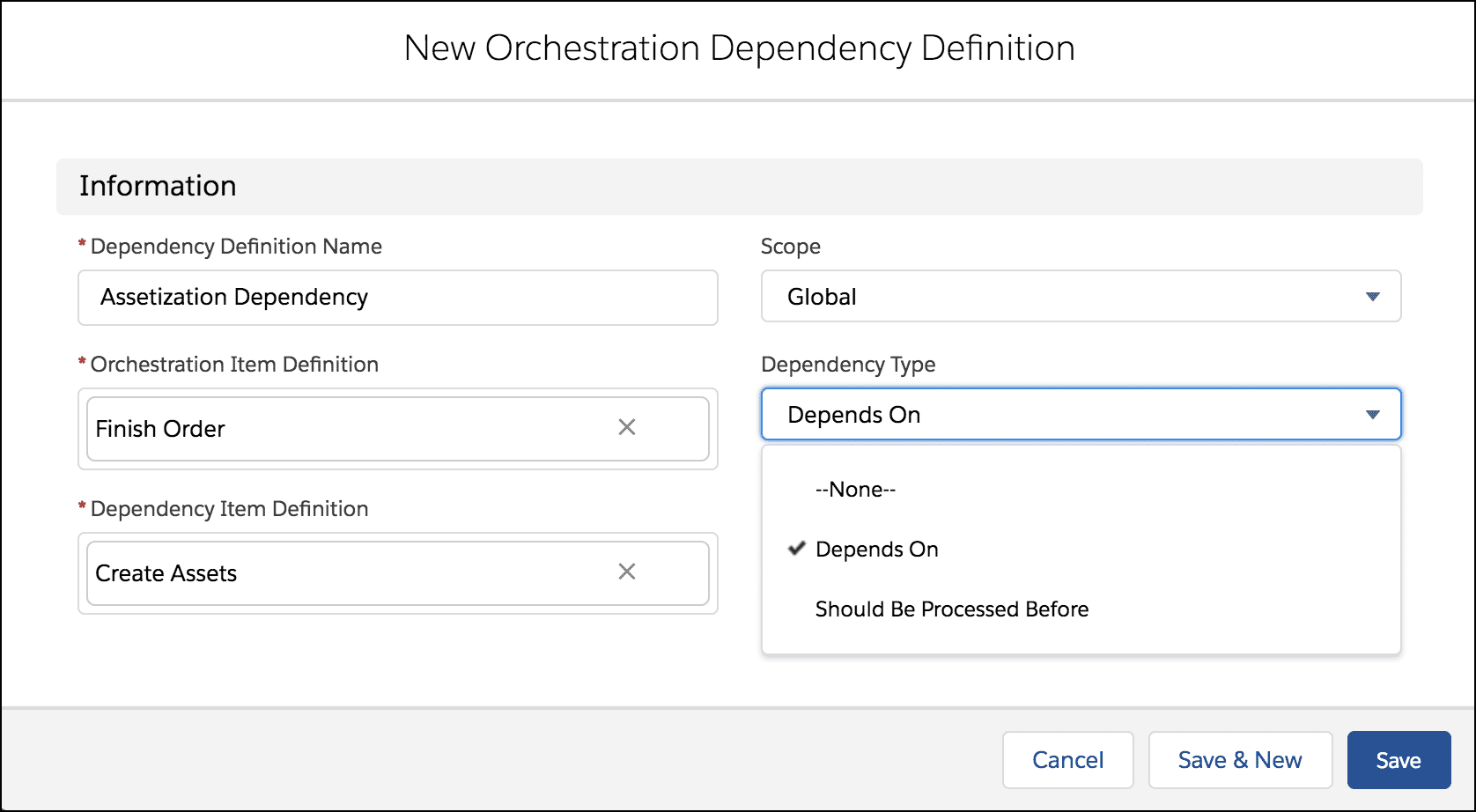
Once assetized, assets display in the asset management lists on the account page. For example, Carole White's assets are shown below.



**Dependency Definitions**  
Sequencing matters in order orchestration as well. Dependency definitions are the mechanism used to specify dependencies between Orchestration Items. You can set dependencies between items in the same plan, or between items in different plans. As you might expect, dependencies enforce the completion of one task before the next task is started. Hence, tasks can be run synchronously, not asynchronously. In short: Complete <Task\_X> before starting <Task\_Y>.

**Creating a Dependency Definition**

Navigate to the **Orchestration Item Definition** within the Orchestration Plan Definition, switch to the **Related** tab, click **New,** and fill out the dependency information.



When setting up dependency information, don't forget to switch to the **Related** tab of the Orchestration Item Definition. (It's easy to overlook this at first.)

Note that there are two different types of dependencies:

1. **Depends On** (default) - Makes the orchestration item depend on the successful completion of another orchestration item. In several images shown above, the Complete Order item (Orchestration Item Definition) depends on the completion of the Create Assets item (Dependency Item Definition). Select the item further down the sequence, and set it to depend on the item before it.
2. **Should Be Processed Before** - Functionally the same as Depends On, but from a different vantage point. The orchestration item must complete before beginning the next item in the sequence. Select the item further up the sequence, and set it to depend on the item after it.

|  |  |
| --- | --- |
| 📝 | Dependencies between items in different orchestration plans do not trigger the execution of the dependent items. That still depends on the proper configuration of an Orchestration Scenario. For example, a task in swim lane B that is dependent on a task in swim lane A, will not automatically execute when the task in swim lane A is completed. *Dependencies don't imply execution.* |

Ultimately, order management is about proper order fulfillment. Salesforce Industries Order Management alone does not accomplish this. It takes third-party systems1 to help complete order fulfillment. Hence, IOM needs to communicate with downstream systems for functionality that extends past our Industries Order Management functionality. For example, communicating with billing, inventory, scheduling, and logistics systems. These are referred to as orchestration systems.

Orchestration systems are used to define external systems that participate in the orchestration process for order fulfillment.

This section covers how to configure Industries Order Management to communicate with orchestration systems. Industries Order Management uses **Callouts** to communicate with orchestration systems. As you learned earlier, Callouts are a type of Orchestration Item Definition used in orchestration plans.

**Creating an Orchestration System**  
Because both "core" Salesforce Industries and Industries Order Management require setup for external orchestration systems, there is a bit more configuration required to set up Callouts as part of an orchestration plan (as opposed to other orchestration items discussed thus far). Let's see the high-level steps first, then drill down deeper from a configuration and then orchestration perspective.

1. Create the orchestration system in Industries Order Management
2. Create the System Interface for the orchestration system
3. Setup the Remote Site Settings in Salesforce
4. Create the Callout Orchestration Item Definition
5. Test the new orchestration system

**Configuration**

Both Industries Order Management and Salesforce need to be configured to communicate with an external system.

**Create the orchestration system in IOM**  
Creating a new orchestration system only requires two pieces of information.

1. **System Name**: Name of the system (e.g. Training\_Mocker)
2. **URL**: URL for request/response communication with the orchestration system. (https://callouts-mock.training.vlocity.xom.vloc-dev.com)
3. Well, maybe three pieces if you include how to invoke the **New System** dialog. Navigate to the **Systems** tab of Industries Order Management, then click New.

**Create thy System Interface**  
Including "thy" in the sub-heading above was an honest typo, but it just felt right to leave it be. It has an air of importance about it.

Although the orchestration system has been created, there is still more information needed to configure two-way communications with it. Switch to the **Related** tab of the orchestration system to create a new System Interface. In addition to a System Interface Name, you must provide:

* **Implementation** - The Apex class used to communicate with the interface. The most commonly used is the **DefaultSystemInterface**, which is a standard interface provided out-of-the-box by IOM to generate the JSON and submit it to the specified endpoint. Note: There are other Implementations, such as XOMAsyncSystemInterface, but the training labs all use the default system interface to communicate with the training mocker.
* **Path** - The Path is appended to the orchestration system URL (specified earlier) to create the final endpoint URL. The communications endpoint for the bulk of the payload between IOM and downstream system communication is the URL provided earlier. However, the URL/Path often contains status information for communication between the systems. For example: {"status":"ok"}

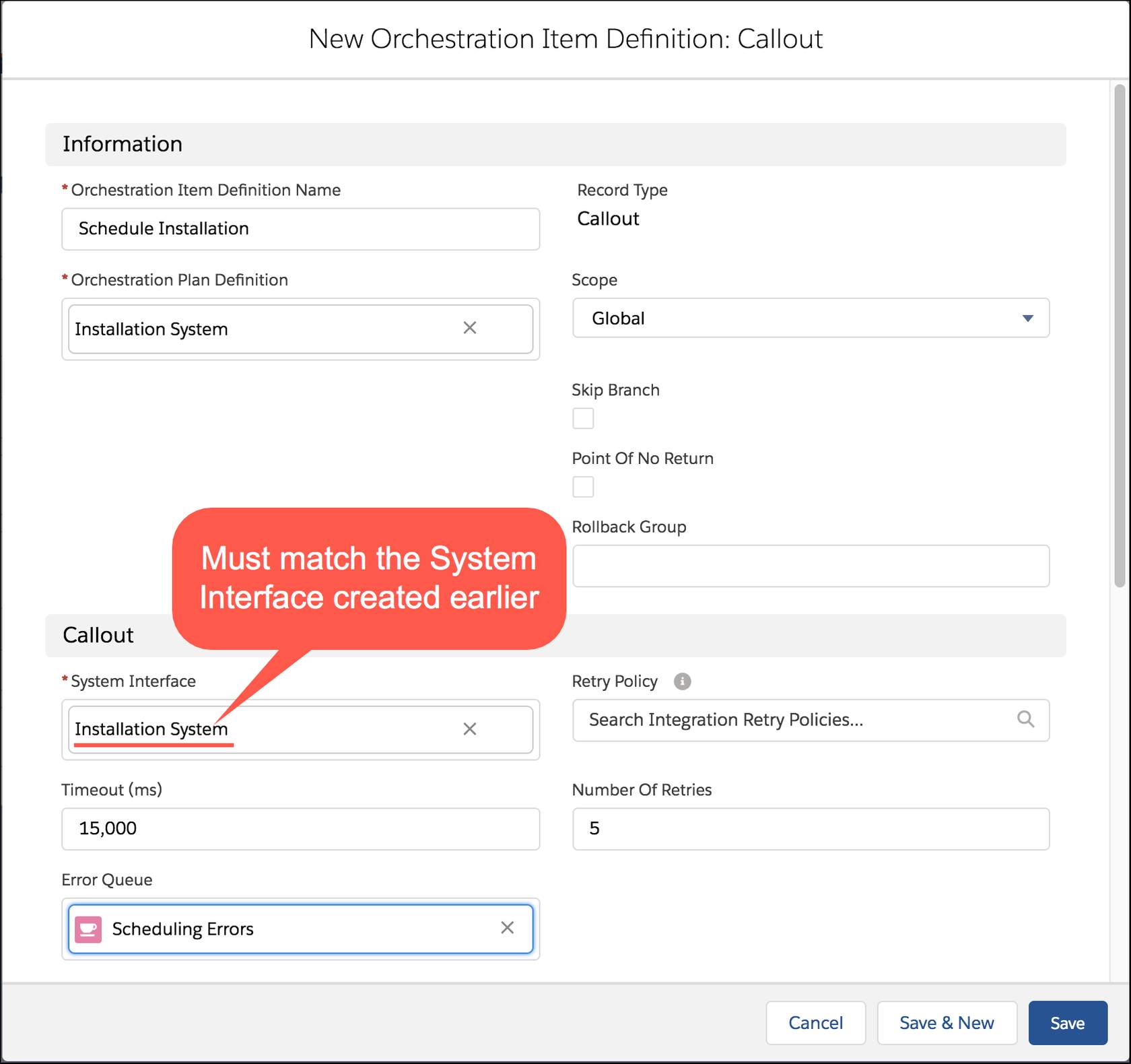
**Set up the Remote Site Settings**  
As you might imagine, the Salesforce platform requires configuration in order to communicate with external systems. This is done from the **Setup** menu in Salesforce.

The good news is that the information needed by Salesforce for a remote site is the same information that Industries Order Management needs for the orchestration system. Navigate to **Setup > Security > Remote Site Settings** and provide the following information.

* **Remote Site Name** (Training\_Mocker)
* **Remote Site URL** (https://callouts-mock.training.vlocity.xom.vloc-dev.com)

**Orchestration**

With the system interface and remote site settings configured, you can get back to the task at hand: Adding a Callout to the orchestration plan. Recall that a Callout is a type of Orchestration Item Definition. As such, it could be added to an Orchestration Plan Definition that already exists, or a new one. Don't forget that if it's added to a new plan, then an Orchestration Scenario is needed to trigger execution.



**Creating an Orchestration Callout**  
When creating a new orchestration callout, there are additional settings to control how the interface behaves during its call and response to the external system.

Several fields warrant further explanation.

* **System Interface** - This must match the System Interface created earlier.
* The next three fields are for error handling and work in concert with each other.
  + **Timeout (ms)** - Set the timeout in milliseconds. If left blank, the default of 30,000 (30 seconds) is used.
  + **Number Of Retries** - Set the number of connection retries to attempt before failure. If left blank, the default of 3 is used.
  + **Error Queue** - Although optional, you can specify a Manual Queue. Callouts that reach a fatally failed state are added to the specified queue. If no queue is provided, you may still see an entry in the Execution Log, but an entry will not be deposited into a queue for processing.

**Orchestration Scenarios** are used to trigger the execution of a swimlane within an Orchestration Plan. They are not visible when viewing the dynamically generated Orchestration Plan. That is why they are depicted as a curved arrow starting external to the plan and ending in the swimlane itself.

**Key Features of Fallout Management**

Industries Order Management (IOM) has several features that address fallout management. The heart of fallout management revolves around what IOM should do when a callout to an external fulfillment system fails.

Notice the fine print? When a callout fails, not if a callout fails.

**Communication breakdown!**

 In the same way communications between people aren’t 100% effective, communications between IOM and external fulfillment systems aren't either.

Integration Retry Policies  
Integration Retry Policies is a powerful feature and offers the most control over how IOM behaves when callouts fail. There are several different types of retry policies that are supported.

New Integration Retry Policy

1. Monotonous Forever Retry Policy - Retry the failed task forever, until it succeeds or is stopped.
2. Monotonous Retry Policy - Retry the failed task a set number of times with an interval between retries. You specify the number of attempts and the interval (in seconds).
3. No Retry Policy - With no retry policy in place, you can still specify:
   * Number of Retries (A finite number, with no interval between attempts.)
   * Error Queue (Manual queue entry if all retries fail.)
4. Staggered Retry Policy - A failed task is retried at the intervals, in seconds, that you specify in a comma delimited list.

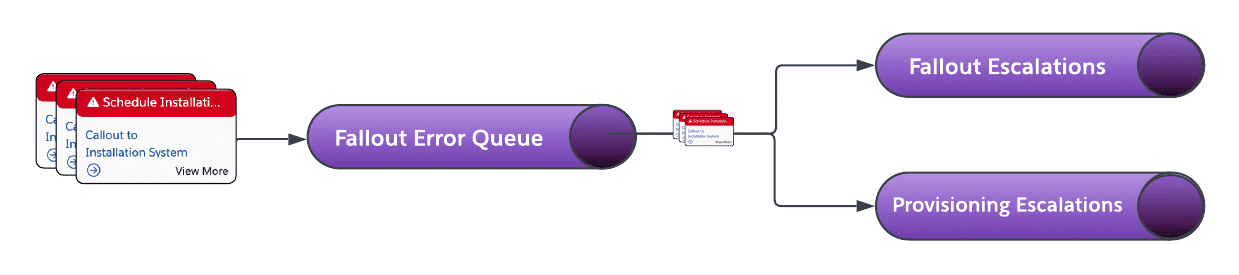
For example: 60, 120, 240, 480, 960, 1920, 3840. Specifies a simple binary exponential backoff interval. Sounds techy, almost nerdy - but put plainly, IOM will retry after a minute, then double that interval and retry again after 2 minutes, then 4, up until about 1 hour between retries before it gives up. This example doubles the interval between retry attempts, but you can enter any number of seconds separated by commas you want.

When configuring a Staggered Retry Policy, you do not specify the maximum number of retries. That is dictated by the number of entries in the comma delimited list.

Retry policies are the focus of this course, but other fallout management features are also covered.

* Manual Queues as Fallout Error Queues
* IOM behavior for different HTTP response codes
* Execution log
* System Interface status (online/offline)

Manual Queues  
Manual Queues are introduced elsewhere in the IOM learning path (Creating Orchestration Manual Tasks and Manual Queues). The CME package install/upgrade process automatically creates a Manual Queue for fallout management, named Fallout Error Queue. Pretty original, huh? Well, perhaps not, but effective, not creative, was the goal with respect to the queue name. It’s common practice to use a regular queue for various callout errors, and then additional queues for escalations. For example, escalate callout failures to an escalations queue that management looks after, or a subject matter expert for a specific downstream fulfillment system.



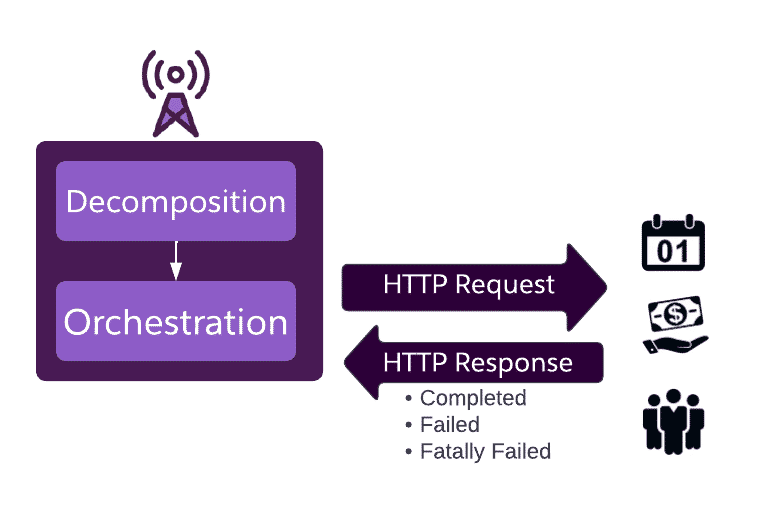
Use the Assign To Queue feature to move entries in one queue to another queue.

HTTP Responses  
At the end of the (fallout) day, what specifically tells IOM there are issues communicating with fulfillment systems? It’s all in the HTTP response received from the fulfillment system.

IOM does listen to the HTTP response codes sent from fulfillment systems. They are mapped to one of a few statuses for the callout.

1. **Completed** - HTTP success code (200, 201, 202,...206). The callout remains in the running state.
2. **Failed** - System-recoverable error (500, 502, 503, 504). A retry may be completed next time.
3. **Fatally Failed**- Error is manually recoverable only. Retries are not attempted.

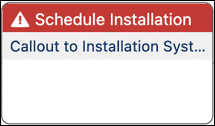
|  |  |
| --- | --- |
| 📝 | The HTTP response codes above are examples only, not an exhaustive list. Please see the documentation in the References section for more information. |



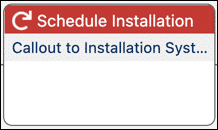
IOM evaluates the HTTP response codes from downstream systems (such as scheduling, billing and provisioning systems).

|  |  |
| --- | --- |
| 📝 | IOM listens and responds to HTTP response codes in the manner described out of the box (OOTB). It is possible to implement your own error-handling routine in a custom system interface. Custom error-handling is beyond the scope of this lesson. |

**Execution Log**  
Additional information on Failed and Fatally Failed callouts is only a right-click away. Select the View Record option for the failed callout and look at the Execution Log field.



Fatally Failed



Failed but retrying

The execution log contains helpful troubleshooting information even if the callout does not have an associated integration retry policy.

4/7/2022, 10:41 AM FATAL Orchestration Item is over the maximum retry limit.

4/7/2022, 10:41 AM FAILED Item is picked up by retry job at 10:41:09

4/7/2022, 10:39 AM FAILED Item is queued for pick up by retry job from 10:40:04

4/7/2022, 10:39 AM FAIL Exception happened during items processing: SystemInterfaceRecoverableException:[errorId=null]: Callout exception System.CalloutException: Unauthorized endpoint, please check Setup->Security->Remote site settings. endpoint = https://callouts-mock.dev.vlocity.xom.vloc-dev.com/billing/activate: trying to retry

**System Interface Status**

There are scenarios where communication between IOM and fulfillment systems will not work - so why even try?

* Planned maintenance window
* Known network communication problems that need resolution

Repeated failed communication attempts can eat up network bandwidth and clog up fallout error queues. Whether scenarios like these are planned or unplanned, you can take the System Interface used by a callout offline.

Administrative Tasks  
Industries Order Management (IOM) has a handful of administrative tasks you should be familiar with. Administrative tasks are found either in Salesforce Setup, or from the Vlocity XOM Administration tab.

Checking on Platform Events Mode

Two key fallout features require that Industries Order Management is in Platform Events mode:

1. Integration Retry Policies
2. System Interface offline mode

Initial releases of IOM used Batch Apex mode. All new installations use Platform Events mode, which offers performance gains when processing orchestration items and is more reliable than Batch Apex mode. Like your hybrid car whose engine transitions seamlessly back and forth between combustion and battery-powered, you can switch back and forth between Batch Apex and Platform Events mode. However, feature availability is impacted when you do.

It’s helpful to know how to check the mode your platform is running in.

In Salesforce Setup, the name/value pair for Orchestration Mode in the XOMSetup Detail of Custom Settings determines the mode your platform is currently running in.

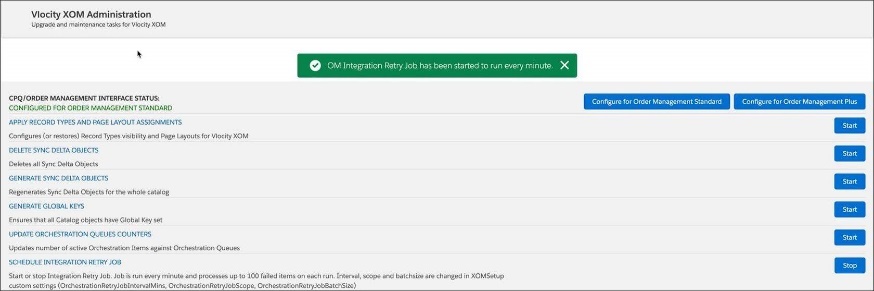
OrchestrationMode must equal PlatformEvents to take advantage of Integration Retry Policies and setting the System Interface status to online or offline.

**Schedule the Integration Retry Job**

Integration retry policies certainly support trying, and trying again. However, they do require a one-time administrative task: scheduling the retry job to run.

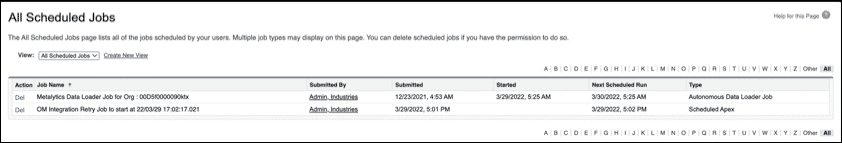
Once scheduled to run, under normal operating circumstances you will not need to schedule it to run again.

To schedule the retry job to start (or stop) running, use the Vlocity XOM Administration page.



Setting the System Interface used by callouts to interact with third party order fulfillment systems also requires the Integration Retry Job to run.

Confirm the Integration Retry Job is Scheduled to Run  
After installation or an upgrade, or perhaps during troubleshooting efforts you may lose sight of whether the retry job is scheduled to run or not. Checking to see if it’s scheduled to run is simple. In Salesforce Setup, look at the Scheduled Jobs.



The integration retry job runs every minute by default. So if you check your system time, the OM Integration Retry job is likely scheduled for the next minute. Look for the retry job’s Next Scheduled Run time.

Configuration Settings for the Integration Retry Job  
You may have noticed the fine print on the XOM Administration page that the retry job has three configurable settings.

* **OrchestrationRetryJobIntervalMins** - Retry batch job frequency in minutes (default is 1)
* **OrchestrationRetryJobScope** - Total number of orchestration items to process with each run (default and maximum is 100)
* **OrchestrationRetryJobBatchSize** - Number of orchestration items to process in one execution cycle (default is 50)

So, for example, to have the Schedule Integration Retry Job run every 5 minutes instead of every minute, change the OrchestrationRetryJobIntervalMins in XOM Setup custom settings from 1 to 5.

Create the Policy  
Launch the Integration Retry Policies application from the App Launcher to create a new policy.

Once created, a policy can be used by any number of callouts in many different orchestration plans. To summarize:

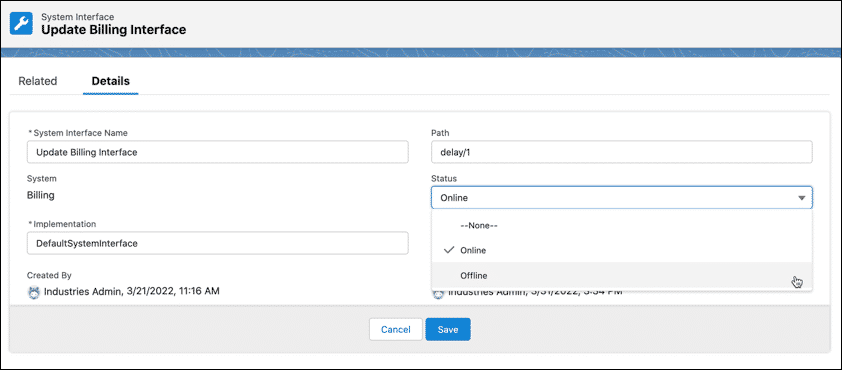
1. *Create* the Integration Rety Policy once
2. *Configure* or modify the Integration Retry Policy over time as needed
3. *Associate* and use the Integration Retry Policy as often as you like

All-in-one Method  
In some scenarios, you may find it useful to consolidate the process and create, configure, and associate the retry policy in a single step. For example, if you know you need a new policy and what callout will use it, you can streamline the process. Edit the Retry Policy for the callout, and use the “+ New Integration Retry Policy” option. This allows you to create the new policy, configure it, and associate it with the callout you are editing in one fluid process.

**Communicating with Fulfillment Systems**

When a fulfillment system is unresponsive, you can set the relevant System Interface offline.

Configuring the System Interface  
The Status field for the specific System Interface can be toggled between Online and Offline.



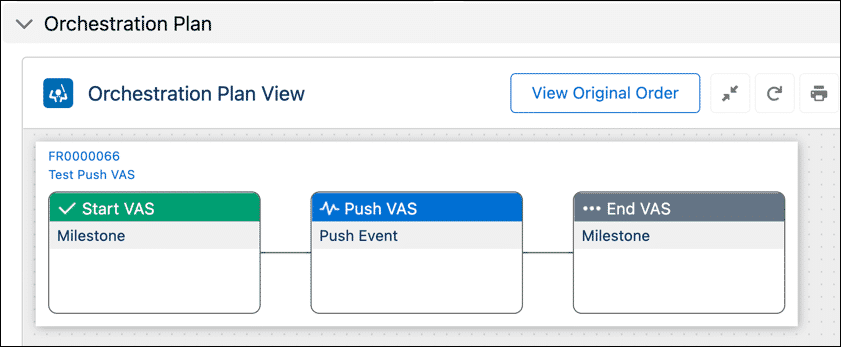
When offline, the status of tasks that use the interface are placed On Hold. The tasks are queued up to be processed later when the interface is placed back online.

**Troubleshooting**

If a callout task remains on hold even after the System Interface it uses is placed back online, it’s usually due to:

* You are a bit antsy and checked the state too soon. (Perhaps it’s that extra cup of dark roast coffee? Or the extra shot of espresso? Take a deep breath, wait about a minute, and refresh your browser.)
* The OM Integration Retry Job is not scheduled to run. (See *Administrative Tasks for Fallout Management* covered earlier.)

**Push Events**  
A Push Event pauses execution until an event evaluates to true.



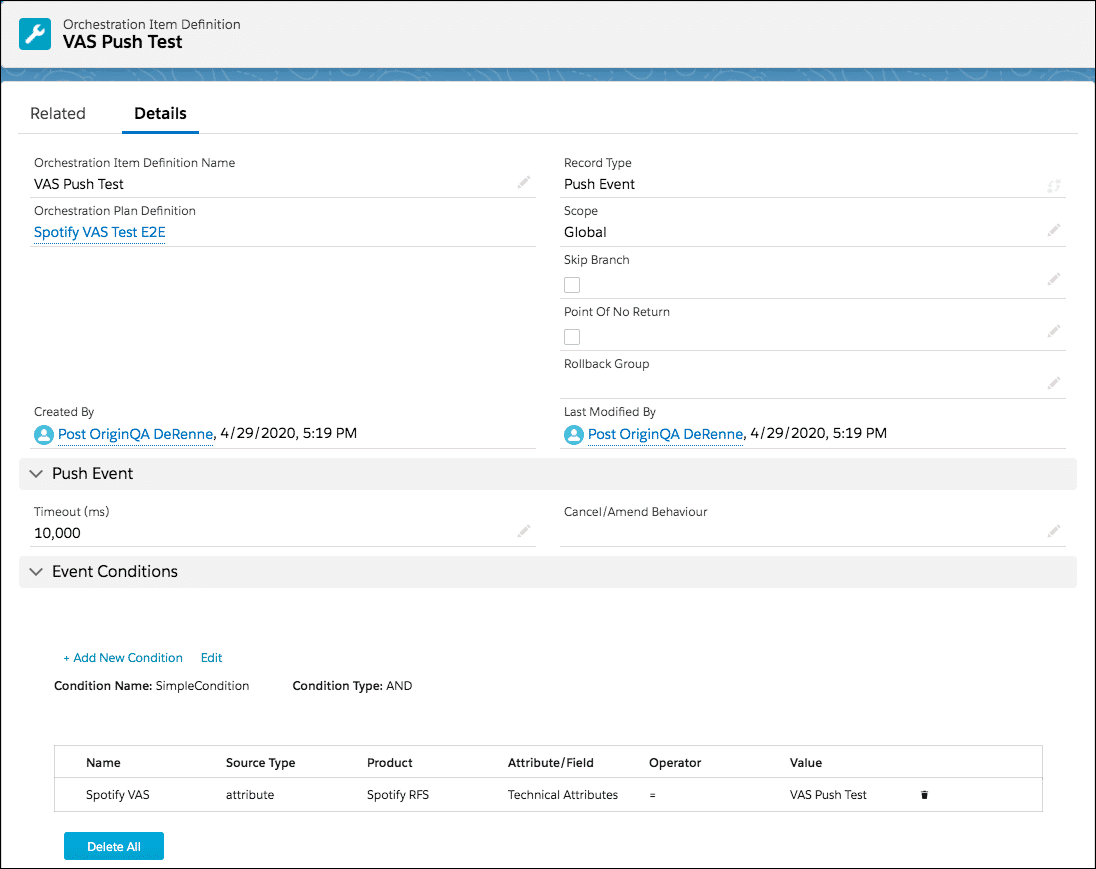
Running Push Event (Push VAS)

Sequence of events

1. **Start VAS** Milestone completes (Status = **Complete**)
2. **Push VAS** Push Event runs and is Waiting for event (status = **Running**)...
   1. ... until its event condition evaluates to true
   2. *then* Status changes to Complete (not shown)
   3. **End VAS** Milestone runs, status changes from **Pending** to **Complete** (not shown)

**Event Condition**

* The Push Event pauses execution until the Event Condition (or conditions) evaluate to true.
* If there is no Event Condition then it evaluates to true (and the task state changes to Complete).
* Event Condition logic is the same as Orchestration Scenario conditions and Orchestration Item Definition conditions.



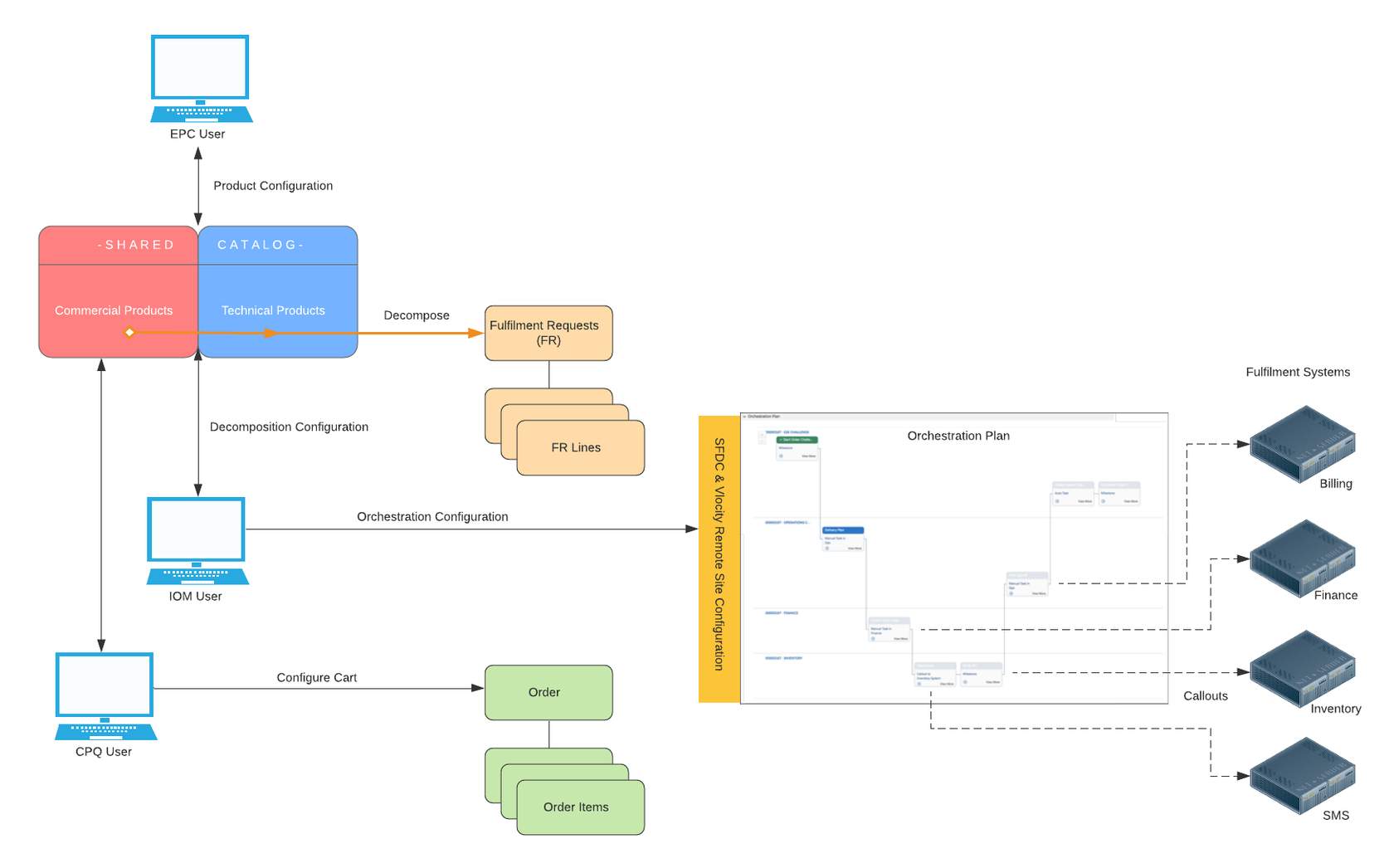
**What types of events trigger a push?**  
Push Events are implemented using Salesforce triggers. Triggers are an excellent means for automation within Salesforce applications, but not all objects are configured for triggers, however. That would be grossly inefficient. What are configured for Push Events?

* **Field** of an Order Item or Fulfillment Request Line
* **Attribute** of an Order Item or Fulfillment Request Line

|  |  |
| --- | --- |
| 📝 | The manner of change to a field or attribute is no concern from an Industries Order Management perspective. A manual edit, Salesforce script/Apex code, integration with another system, API - it simply does not matter. |

**What is MACD?**

* **M**ove - Move products or services to a new location
* **A**dd - Add new products or services
* **C**hange - Change the configuration of existing products or services
* **D**elete - Delete existing products or services



**MACD Decomposition**

**Decomposition Logic**

notice that the one-to-one and one-to-many decomposition models end up with different fulfillment request lines (FRLs) for the same source order action (Modify).

| **Model** | **Source Order (Action)** | **Fulfillment Request Line (FRLs)** |
| --- | --- | --- |
| One-to-one | Modify | Modify |
| One-to-many | Modify | Disconnect (old service)  Add (new service) |

Industries Order Management looks at current assets for the account, then performs a diff1 of the "as is" state and the "to be" state after decomposition. When decomposing, Industries Order Management factors in the product model, decomposition relationship, including any conditions (such as the download speed technical attribute in our example). In so doing, it determined that both Disconnect and Add actions were required in the decomposition results (FRLs). To summarize, the actions you see in CPQ do not necessarily translate directly into decomposition results in Industries Order Management. OM does not blindly take actions from CPQ... under the hood there is a level of complexity that is going on to ensure correct decomposition results.

There are of course additional product models, decomposition models (with varying conditions), actions, and orchestration plans that can be modeled and tested. (Some are covered in more detail in upcoming hands-on lab exercises.)

1 diff is short for differentiation. A diff operation compares two objects and returns the different results. (That is, what actions would it take to make the two objects equivalent.)

**MACD Orchestration**

Once the shared catalog and decomposition relationships are configured for your use cases, the focus of MACD with respect to Industries Order Management turns to orchestration. Recall that orchestrations are triggered by the orchestration scenario. There are three key components when configuring an orchestration scenario:

1. **Product**
2. **Action**
3. **Condition**

**Key Terms**

* **In-flight order** - An order that has been submitted (from CPQ to IOM) but not completed.
* **Supplemental order** - An order created by CPQ to revise an in-flight order. Supplemental orders supersede the original in-flight order.
* **Point of no return (PONR)** - Point in the order orchestration process that once passed, order cancellation is no longer possible or permitted.

**Salesforce Status**

The Salesforce order Status field is a picklist with two values by default. From an Industries Order Management perspective, you will observe the following for the Salesforce Status field:

Draft - The Salesforce Status setting from the initial order capture, through CPQ submission to OM, and during the entire order fulfillment process remains Draft.

Activated - The Salesforce Status setting transitions from Draft to Activated when the order fulfillment process is either completed or canceled.

**IOM Order Status**

Industries Order Management requires extended functionality, so an additional Order Status field is used to support very specific work flows.

The All Orders view of the Orders tab for the training playground has been setup to include the Salesforce Status, Order Management Order Status, and the Superseded Order fields. This provides critical status information at a glance, which is helpful when learning in-flight order cancellations. You can also see the Order Status inside the Cart.

**Orchestration Plan & Task States**

The state of tasks within an executing Orchestration Plan has already been covered. However, don't overlook that you can see the overall Orchestration Plan state near the top of the Details tab for the executing plan. The screenshot below shows a plan In Progress. As an example of what was mentioned earlier, when order fulfillment finishes, the State shown here transitions to Complete, and the Salesforce Status changes from Draft to Activated.

*Note*: The Salesforce Status, IOM Order Status, as well as the plan and task states are all observed during the upcoming lab exercise on order cancellations.

**Key Points**

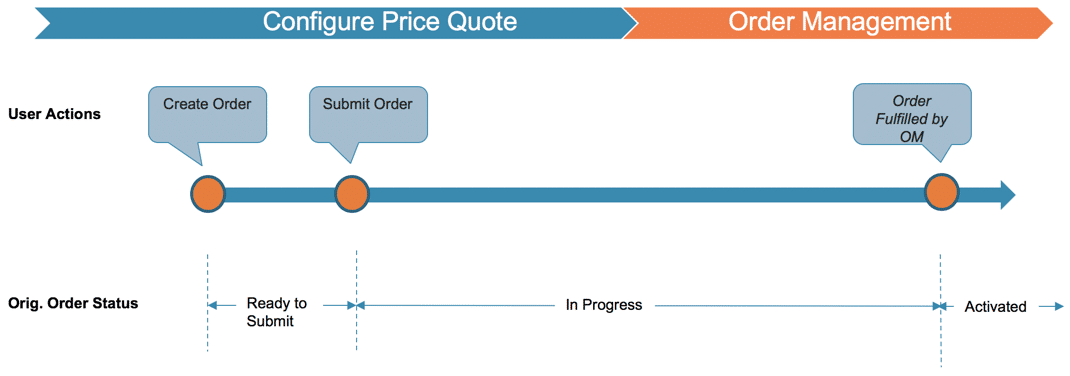
* Order cancellation uses supplemental orders that are automatically generated by CPQ
* Supplemental orders are automatically decomposed as well
* Cancellation is at the order-level or order-item (depending on what is selected on the order and PONR configurations)
* When an order is canceled by CPQ and accepted by Industries Order Management, the **original order**:
  + Status is changed to Cancelled
  + Order is locked
* When an order is cancelled by CPQ and accepted by Industries Order Management, the **supplemental order**:
  + Status is changed to In Progress (and eventually Completed when the orchestration plan finishes)
  + Is read-only mode

**CPQ/IOM Communications - High-Level View**  
The process flows below show Configure Price Quote (CPQ) and Industries Order Management (OM) communications at a high level.

**Standard Order**

This basic process flow diagram shows communication between CPQ and Industries Order Management, including the status of the original order as standard user actions progress (create, submit, and order fulfillment by Industries Order Management).

Note: Although this presentation refers to "User Actions" such as the **Cancel Order** button in the Cart, the process could be initiated via the CPQ API.



**Order Cancellation**

The order cancelation process is a bit more involved.

**User Actions / System Events** – Create, submit and cancel order are initiated by the user, other actions are actually system events automatically triggered based on configuration and prior user events.

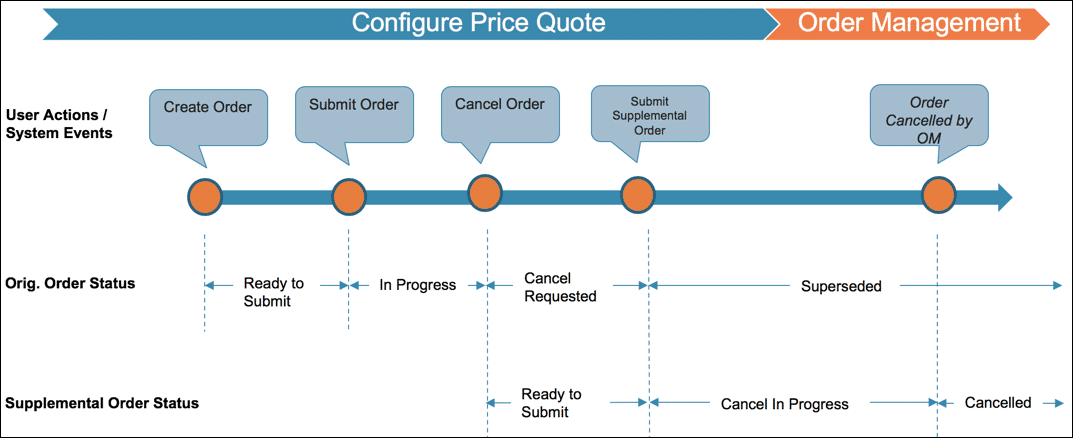
* Create/submit supplemental order (this happens in the background without user intervention)
* Order cancellation and fulfillment by Industries Order Management

**Original Order Status** – Additional order states are required to implement order cancellations. Both CPQ and OM communicate and query the new states.

* The original order transitions from Cancel Requested to Superseded (assuming the order is not past PONR, and the cancel is performed).

**Supplemental Order Status** – The CPQ generated supplemental order needed to implement the cancellation requires new order states as well.

* If all goes well between CPQ and OM during the cancellation process, the state transitions from Cancel in Progress to Canceled.
* Not only does CPQ automatically generate the supplemental order, but it also initiates decomposition as well, and links it to the original order.
* Note that similar to a locked or read-only order, the canceled supplemental order can’t be changed.



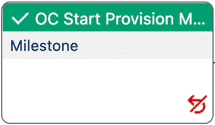
**PONR (point of no return) Details**  
There are a few more PONR related details to understand. The details are widely known throughout the industry as the GAAG of PONR.

* **G**olden Rule
* **A**ppearance
* **A**ssetization
* **G**ranularity

Actually, GAAG isn't widely known throughout the industry. In fact, it isn't known at all, but it seemed like a good time to introduce a new acronym!

**Golden Rule** -  A task in an executing orchestration plan reaches PONR when it advances to any of the following states: **Completed**, **Running1**, **Failed**, **Fatally Failed**.

1How a running task impacts the freezing of an order during a cancel request depends on its "Cancel/Amend Behavior" setting. This setting is also known as Smart Freeze and is covered in an upcoming lab exercise.



**Appearance** - The red looping arrow with a slash indicates the Orchestration Item Definition (e.g. task) has been set as PONR.

**Assetization** -If no PONR is defined, once the order is assetized, **it cannot be canceled**. There is no out-of-the-box way to *un-assetize*. In most cases, you should activate PONR on your Assetization auto-task.

|  |  |
| --- | --- |
| 📝 | If adhering to best practices and your Assetization auto-task occurs as the last non-milestone task in an orchestration plan, then this should not be an issue. |

**Granularity** - Industries Order Management supports cancellation at the order-item level. (The initial implementation only allowed for the entire order to be canceled, not individual items within the order.)

When an order item that is marked as PONR moves to a Running state, then the following state changes occur:

* The order items that are linked to the PONR orchestration item are marked as PONR
* All upstream order items are marked as PONR, all the way to the root order item
* All children of the root order item, and their children, are marked as PONR

The basic diagram below should help clarify granularity pertaining to PONR and canceled orders.

Rollback Plans are for undoing whatever was already completed in an orchestration plan when an in-flight order was canceled. For example, if ports and copper pair were allocated for an order, and the order is canceled... a Rollback Plan can be used to free up those resources.

Rollback Plans are built in the same manner as regular orchestration plans, but they do have a few differences you need to be aware of.

Setting up Rollback Plans presupposes you already have the orchestration assets in place. That is, Orchestration Plan Definitions, associated Orchestration Scenarios and Orchestration Item Definitions.

**Rollback Groups**  
Rollback Groups are the last key concept to understand when it comes to implementing in-flight order cancellations.

* Rollback Groups change the order of execution of rollback swim lanes
* By default, Rollback Groups execute in the same order as the original orchestration swimlanes
* Rollback Groups do not change the execution order of tasks within them. (That remains a function of dependencies.)