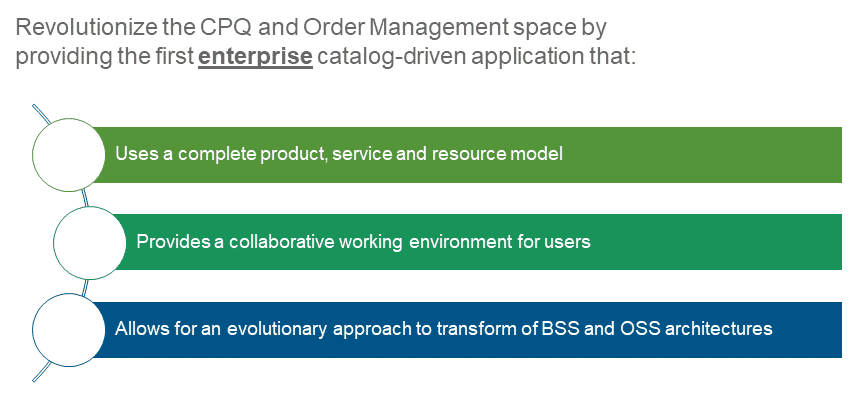
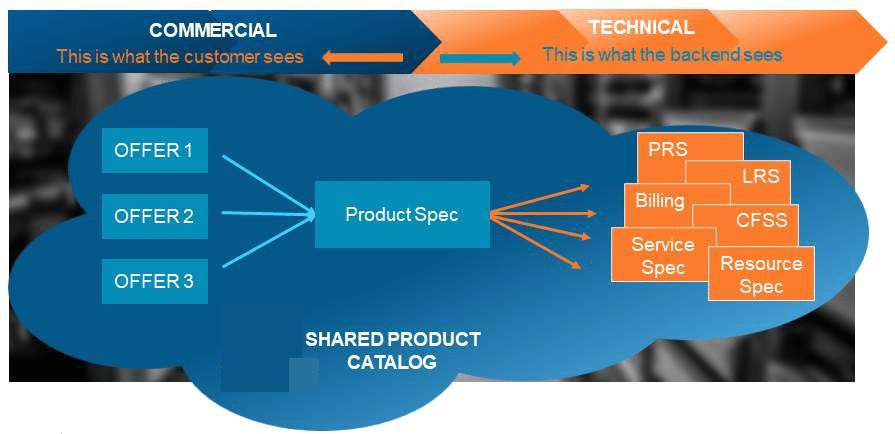
**EPC Overview**

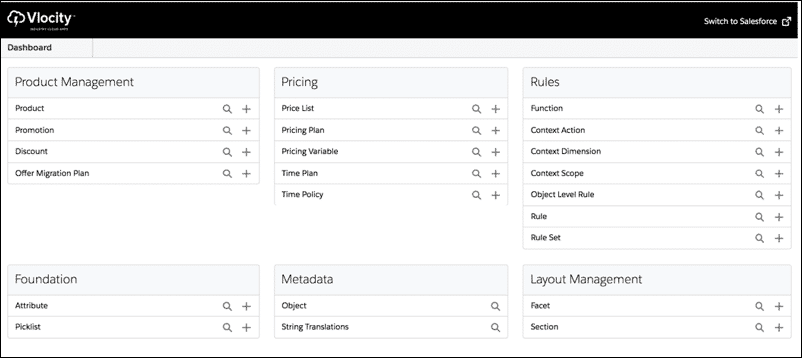
**The Enterprise Product Catalog (EPC)**



**Shared Product Catalog**

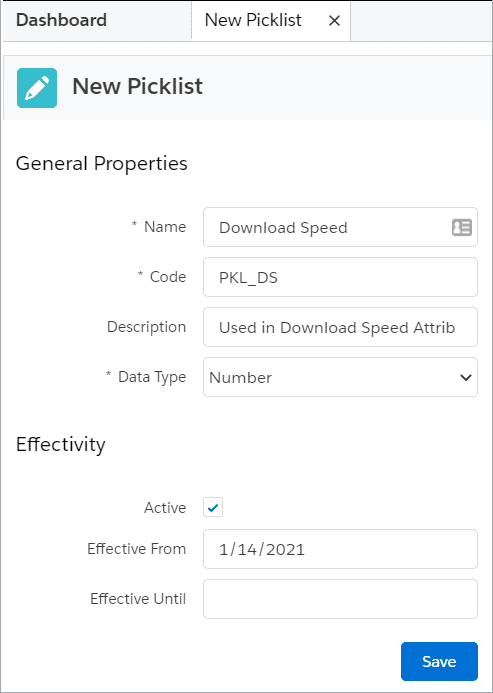


**The Product Console**



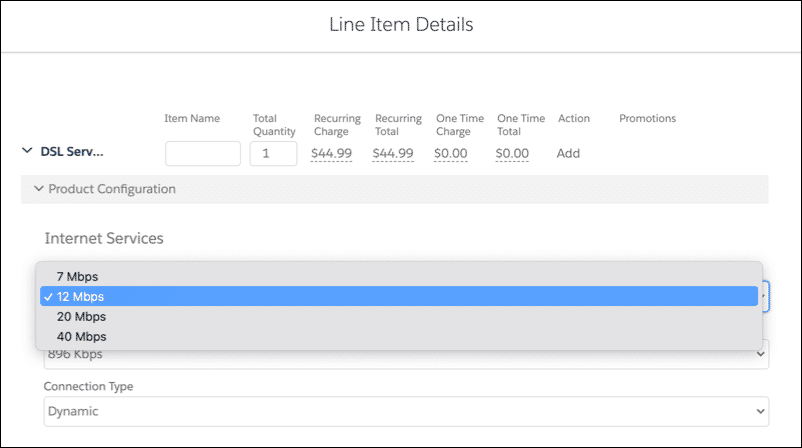
* “EPC Command Center”
* Unified user experience to manage EPC elements in a single, contextual interface
  + Products
  + Attributes
  + Picklists
  + Product Object Types

# About Picklists



* Picklists are designed to be reusable across attributes, products, and offers
* Name, Code, and Data Type properties are required
* The name of the picklist is used only at design-time and not at run-time. This means it is not visible to customers.
* We recommend (but do not enforce):
  + Use PKL\_[code] naming convention
  + Setting the Active flag and Effective From date

# About Product Attributes

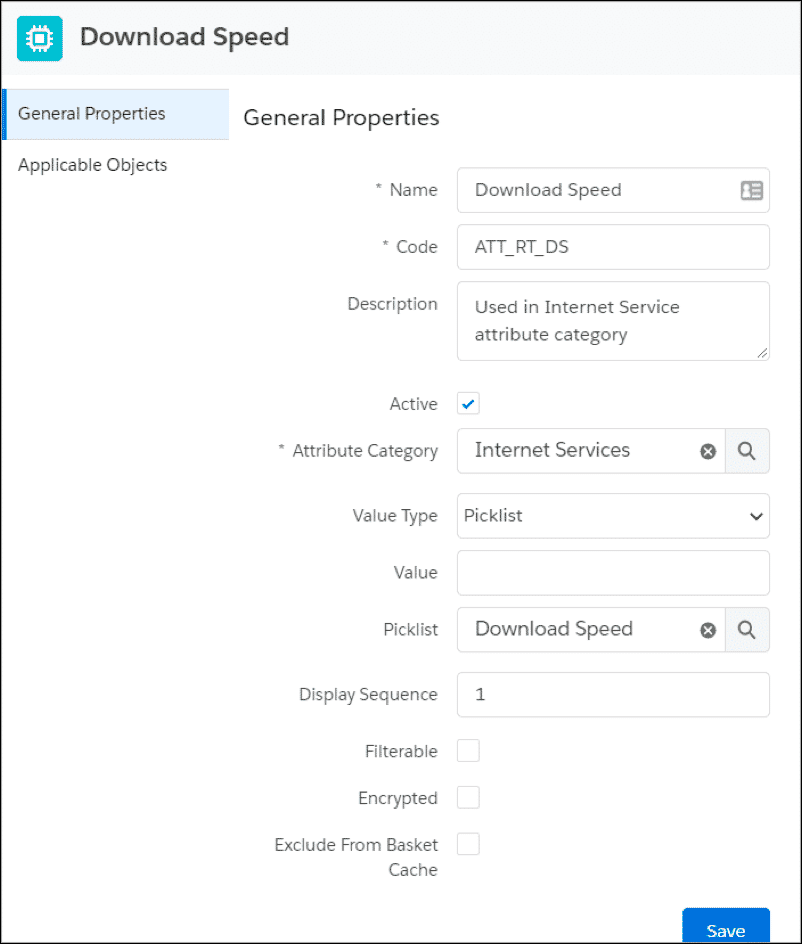


* Provides a powerful and flexible way to extend product entities
* Stored in aggregate “blob” form in the JSONAttribute field on the Product object
* Used extensively throughout Industries CPQ and Industries Order Management (IOM) to:
  + Filter products at run-time
  + Configure product specifications at run-time
  + Change product pricing using attribute-based pricing rules
  + In Industries OM, to map commercial products to technical products

**When to Use An Attribute Versus a Field**

|  |  |
| --- | --- |
| Field | Attribute |
| If the data element is common for all products across the entire product catalog, create a new field on the Product object.  Typically, adding new fields requires Salesforce Administrator privileges. | If the data element is specific to a product or a class of product, create a product attribute.  OOTB, attributes are used at run-time (visible to customers) in the Cart to filter the product catalog and to provide product configuration capabilities.  Adding attributes simply requires access to the Product Console. |

**Product Attributes - General Properties**



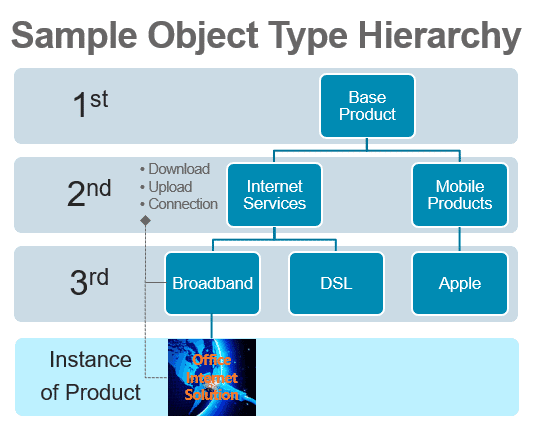
* The **Attribute Name** is required and visible at run-time so it's important to pick a customer friendly name.
* The **Attribute Code** is required but not visible at run-time. We recommend the following naming code convention: ATT\_[code].  This makes it easy to identify the EPC element type during design-time.
* The **Attribute Category** is required and is created through Attribute Categories in the Product Console.
* You can optionally designate a **Picklist**to provide the values for your attribute.
* **Filterable** attributes can be used within the Cart to filter product lists and order items.
* You can set your attribute to be **encrypted**, if it contains sensitive information.
* Product Attributes can be assigned directly to a product or to a product Object Type, which allows it to be inherited.

**Exclude from Basket Cache**should be applied only to attributes that do not affect pricing or configuration validity.  Attributes with this flag are ignored in API requests in relation to pricing or configuration, reducing overhead and thus improving performance.  This feature is not used within the CPQPartition platform cache, but instead is used with caching either on the Salesforce platform or on the Digital Commerce Gateway.

# What is a Product Object Type?

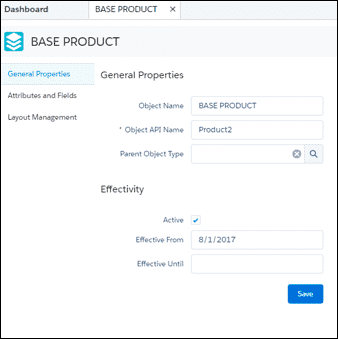
* Reusable entity that defines properties (fields and attributes) and layout for all product instances
* Used to group products with similar characteristics and ensure consistent behavior and application of rules
* Supports hierarchy inheritance
* At this time, we do not recommend moving object types within the hierarchy. Careful planning is required during initial setup.
* Salesforce Industries Expert Services can assist with product catalog design and modeling

**Object Types "Is A" Inheritance Architecture**



* Object Type is the 1st level of abstraction or “Supertype”
* 2nd+ levels of abstractions are “Subtypes”
* The application of the Object Type is an “instance-of” relationship, or Product Instance

**EPC - Product Object Types**



* We recommend creating a “Base Product” during initial setup. This will be your product catalog’s “primordial object” and should include all of the fields in the product catalog.
* Object types for specific product classes can then be created to inherit the Base Product’s layout, fields and any attributes.
* The Base Product’s layout must be created manually.
* An object subtype will inherit the layout via deep copy, but any subsequent changes to the object super type’s layout will not be replicated down the hierarchy after initial creation.
* Inherited attributes and fields cannot be deleted, and instead must be unassigned from the object super type.
* New attributes and fields assigned to an object type will be inherited dynamically by its subtypes.

# What is a Product Life Cycle?

A product life cycle is the process that every product goes through from its introduction to its retirement. There are four defined stages for a product lifecycle: introduction, growth, maturity and decline.

The shared product catalog supports **commercial** product lifecycles by allowing you to define current, future, and retired products based on product selling period dates.

1. **Current Product**

A product that is currently available for sale.

1. **Future Product**

A product that will be sold in the future. This product can be configured and ordered but will not be fulfilled until its selling period and fulfilment begin.

1. **Past Product**

A product that is no longer for sale but still can be fulfilled for orders that have already been placed and changed for orders that have already been assetized

1. **Retired Product**

A product that is no longer supported, cannot be ordered or fulfilled, or changed for orders that have been assetized.

# Setting Product Selling Periods on Products

Product selling period dates are set in the General Properties facet of the product. The Product Console validates all date and times entered using the rule:

**SellingStartDate  <  =  FulfilmentStartDate  <  =  SellingEndDate  <  =  EndOfLifeDate**

The Product Console will not allow you to save products until the dates are set correctly. If you choose not to enter product selling period dates, it implies that the product is available for all of time.

**Caution**

Product selling period data is stored in the platform cache. After you make changes to product selling period fields, you must update the product hierarchy data in the platform cache in this sequence:

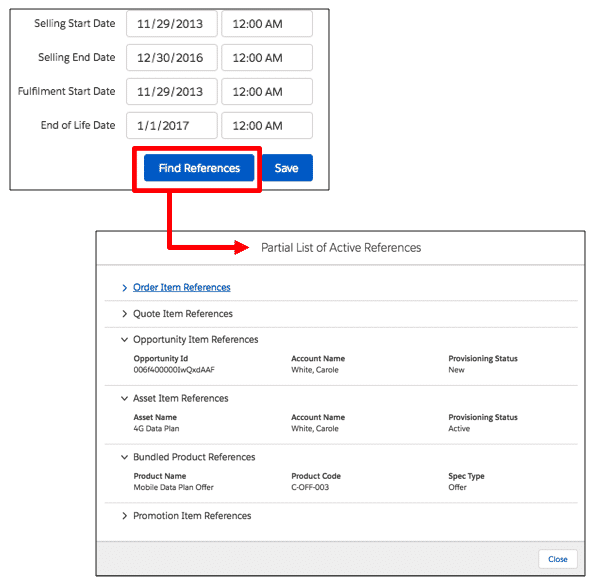
1. Product Hierarchy Maintenance
2. Clear Managed Platform Cache
3. Refresh Platform Cache

When a product is used in bundles, it is the responsibility of the product administrator to ensure the selling periods are consistent.

**Find References**

Product administrators use the **Find References** button to find references to a given product in active promotions, product bundles, assets, quote line items, order line items and opportunity line items. The button assists administrators in determining the effect of changing product selling period and end of life dates.

The default number of rows returned for each reference object is 20, but the limit can be increased by setting the FindReferencesRowCountLimit custom setting in CPQ Configuration Settings Overview. The maximum number of rows returned is 1000.



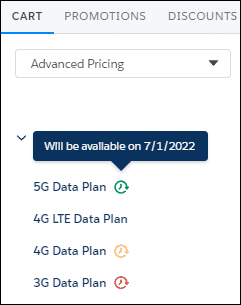
**Cautions**

In order to find order, quote and opportunity line items references, each line item's fulfilment status (vlocity\_cmt\_\_FulfilmentStatus\_\_c) must be set to Draft, In Progress, PONR (point of no return), or Pending.

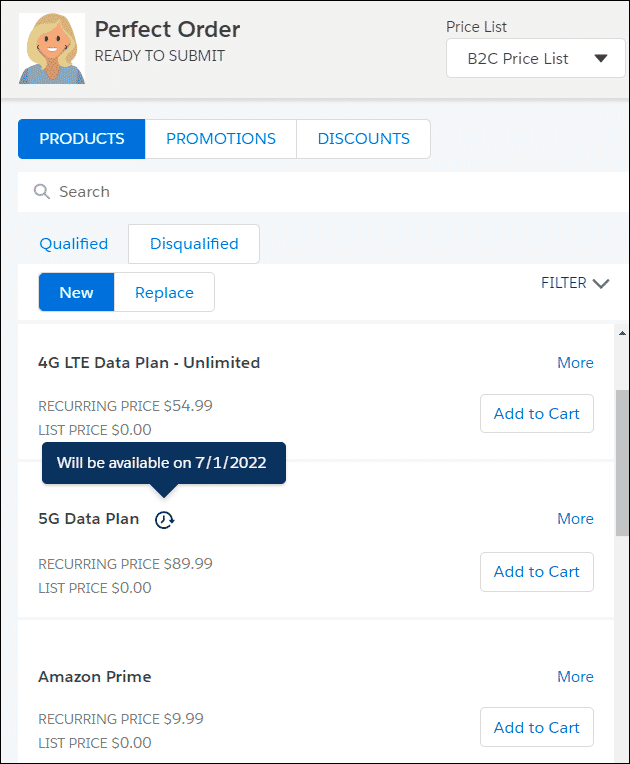
In order to find asset references, the asset's provisioning status (vlocity\_cmt\_\_ProvisioningStatus\_\_c) must be set to Active, Changed or Deleted. These values are normally set by order management processes. If the statuses have null or other values, the references will not display.

**Product Selling Periods in the Cart**

To support MACD and asset-based ordering, the Cart can display current, future, past and retired products based on the Order (or Quote or Opportunity) Start Date



* Current and future products display in the cart normally, but **future** products are indicated with a **green** **clock** icon.
* **Past** products are products that have passed their Selling End Date but have not passed their End of Life Date. They are indicated by a **yellow** **clock**icon. They cannot be added to the cart or configured for new orders, but they can be configured or deleted during MACD or asset-based ordering.
* **Retired** products are products that have passed their End of Life Date and they are indicated by a **red clock** icon. In a new order, they cannot be added to the cart or configured but during MACD or asset-based ordering they can be deleted.



Similar to products added to the cart, the product list also displays current and future products based on the Order Start Date.

**Caution:**Product selling period functionality does not override other standard controls for displaying products in the Products list. In order to display, products must be marked Orderable, Active and have a price list entry. The price list entry Effective From date must be current, as well.

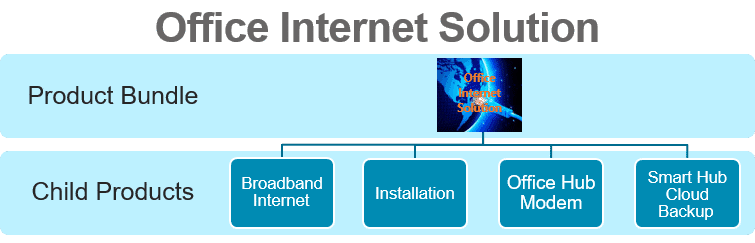
* Products that have reached the end of their selling period or are at their end of life date will not display in the product list and can’t be added to the cart.
* You can add a future-dated product to the cart and submit the order normally; however, Future Dated Ordering functionality must be implemented in order to manage the fulfilment of future products.

# Creating Products and Product Bundles

**EPC Products**

* We recommend (but do not enforce) that all products be created with an object type.
* New attributes and fields can be assigned directly to a product, if desired.
* Attributes and fields inherited from an Object Type cannot be unassigned.
* Commercial products must have a price, be Orderable and Active.
* Products are bundled using a “Has-A” Containment relationship as either a child or a realization.
* We recommend using Specification Type to designate Products, Offers, Resources, and Services.

**Creating Products in the Product Console**



1. Create Office Internet Solution product bundle
2. Set the attributes
3. Add child products
4. Add to price list

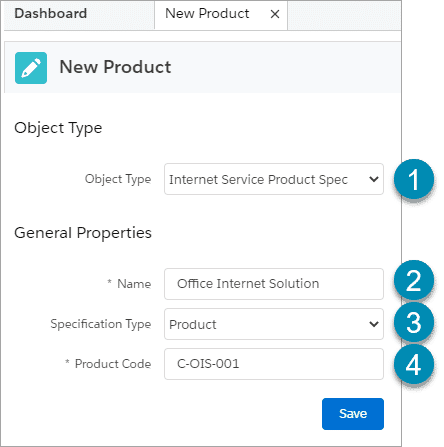
**1. Create Office Internet Solution product bundle**

**The Product Console**

Now, we're ready to create our new product - Office Internet Solution - which is a product bundle.

From the Product Console Dashboard under Product Management, we see Product. Click the **+** sign next to Product to create a new product.

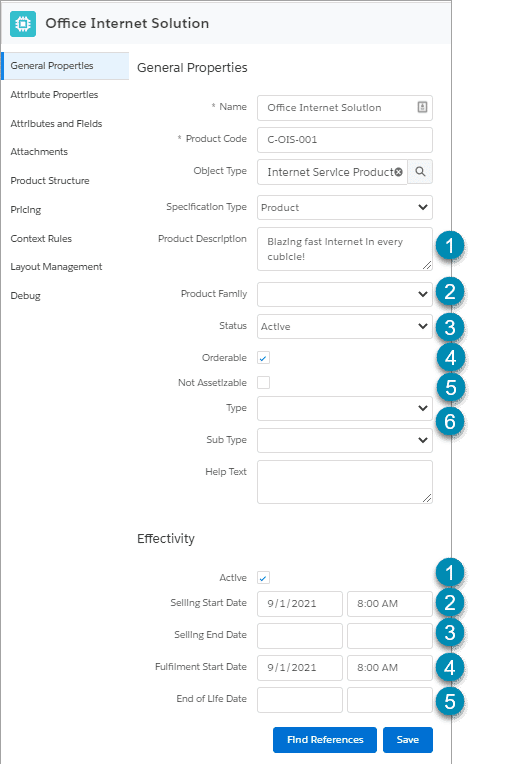
There are two parts to the process. On the **New Product** page, we'll enter the following field information:



1. Set the **Object Type** for the new product.
2. Under **General Properties**, provide a **Name**for the new product.
3. Select the **Specification Type**. In this instance, we'll select Product. Other options include: Resource, Service, Offer or Grant.
4. Assign a **Product Code** to the new product - because it is required.

Click **Save**to continue.

The second part of the process involves completing the remaining information for your product. You will see that the object type, product name, specification type and product code you entered previously has been prefilled on this screen



**General Properties**

1. Give a **Description**of the new product.
2. **Product Family** is an optional field related specifically to your company that allows you to further classify your product.  For example, an energy company may want to specify whether this product relates to gas, electricity or a combination of both.
3. Set the **Status**of the product, we'll select Active. Other options are: Draft, Pending Approval.
4. Check the **Orderable**box so that the commercial product is visible in the Cart.
5. You should only check **Non Assetizable**if the product you are creating should not appear in the Asset list for the customer. For example, if you give a free promotional t-shirt with every order, you would not necessarily want this to appear as part of the customers' assets.
6. **Type**and **Sub Type** are further optional classification fields which may or may not be used by your company.

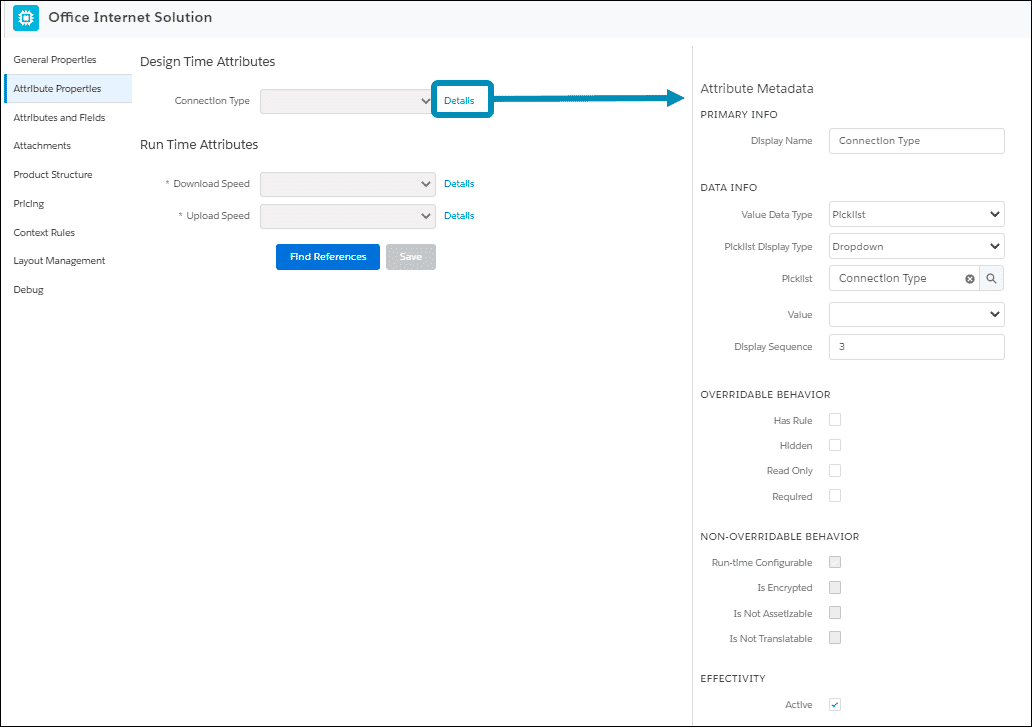
**Effectivity**

1. Check the **Active**box to ensure the product is visible in the Cart.
2. Enter the **Selling Start Date** and time the new product will be available to be ordered and sold.
3. The **Selling End Date**can be defined if you know when the product will no longer be available to be ordered. In this instance, we'll keep it blank.
4. Enter the **Fulfilment Start Date** and time of the new product. This defines when the product is ready to be fulfilled.
5. Enter an **End of Life Date** if you know the date in which the product will no longer be supported. In this instance, we'll keep it blank.

**Save**the product once all appropriate fields have been populated.

**2. Set the Attributes**

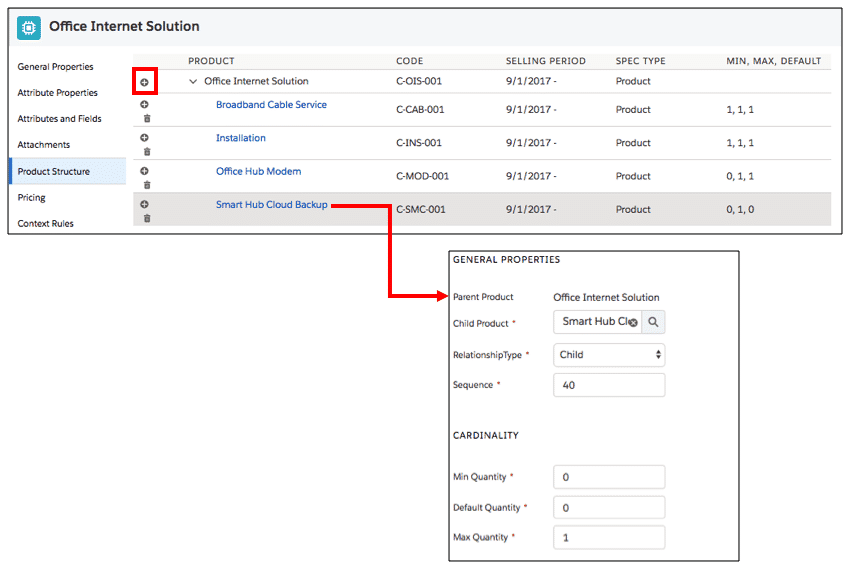
Next, we'll set the attributes for the new product. Click on the **Attribute Properties** facet of the new Office Internet Solution product and amend the attribute metadata, as shown below. When you are done, click **Save** in the **Attribute Metadata pane** to save your metadata.



The same steps would be followed to configure the Run Time Attributes for Download and Upload Speed. However, you'll notice the red asterisks next to Download and Upload Speed and that the fields are grayed out. This is because the Required box is checked in the Overridable Behavior section for these attributes and the Run-time Configurable box was checked in the Non-Overridable Behavior section making the user required to select these attributes in the Cart. Remember, non-overridable behavior is set on the attribute at the object type level.

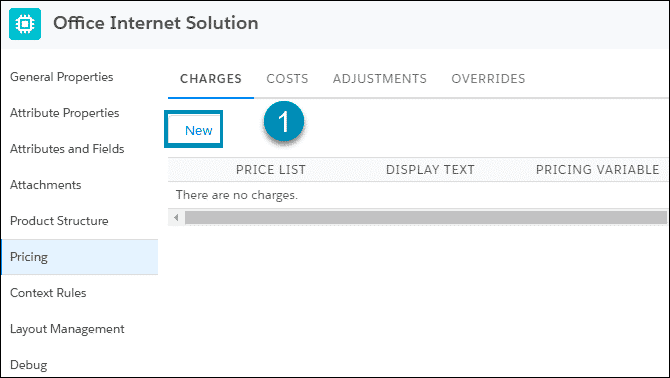
**3. Add Child Products**

Now that the attribute properties have been set up, the next step is to create the product structure for the bundle. Click on the **Product Structure** facet of the new Office Internet Solution product.



**4. Add to Price List**

The last step that needs to be completed to create the product bundle is to give it a price. Click on the **Pricing** facet to assign a price list and price.



1. Click on the **New**button under the **Charges**tab to add new charge details. You will see the **Price List Entry** details appear on the right.
2. **General Properties**
3. Click the search icon next to the **Price List** field to select the price list that will be associated with the bundle. In this instance, the B2B price list is selected.
4. Enter descriptive **Display Text** as this is displayed in the products list in the Cart. In this instance, the bundle will be $250 or more depending on the child products selected.
5. Check the **Base Price** checkbox to display the price of the bundle in the products list in the Cart.
6. **Pricing Variable**
7. Select the **Charge Type** - Recurring in this example.
8. Set the **Frequency** - Monthly in this example.
9. Select the **Sub-Type** - Standard in this example.
10. Click the **Search** button.
11. Select the **Pricing Variable** - Recurring Monthly Std Price in this example.
12. **Pricing Element**

Select the **price** of the bundle. In this example, the price is Free because the price of the child products are rolled up into the total price of the bundle.

1. **Time Plan/Policy**

This relates to when we want the pricing to apply in relation to the order date and how it should be applied.  You will learn all about this in the pricing modules of this course.  For now, please leave the time plan and policy blank.

1. **Effectivity**
2. Enter the **date and time** the pricing of the product or bundle is effective.
3. Check the **Active** checkbox.

Don't forget to click **Save**to save all your changes!

# Managing the Shared Catalog

**What does it mean to manage the Shared Catalog?**

Enterprise product catalogs always require administration and management. In the case of Salesforce EPC’s shared catalog, there are several tasks that must be completed when you are creating or updating elements in the catalog. These tasks are designed to ensure top performance and faster speeds at run-time.

Salesforce Industries has created several administrative jobs to help you complete these tasks. The jobs primarily do three things.

* **Transform** catalog entities into flattened or streamlined records that are easier and faster to read and load.
* **Cache** catalog data into memory rather than having to request data from the database.
* **Regenerate** metadata that is critical for CPQ operations.

When changes are made to the shared catalog, catalog administrators need to determine when to run each job. Some jobs can be run without affecting live production environments, but others, such as the Clear and Refresh Platform Cache jobs, must be run during downtimes or low usage.

For example, if you run the Product Hierarchy Maintenance or Refresh Platform Cache jobs while people are using CPQ, the maintenance jobs can disturb the CPQ APIs. This means that CPQ may produce incorrect data whilst the maintenance jobs are in progress. From the Fall ‘20 release onwards, there is a **Delete Old Data**checkbox beneath the Product Hierarchy Maintenance option, which allows you to clear old data, or keep it. If you choose to keep it, the APIs will run off the old data until the new data is in place, so there is no downtime whilst the batch jobs are running.

These jobs do not impact Industries Order Management order decomposition or order orchestration, and will not affect decomposition or orchestration that is happening as the jobs are run.

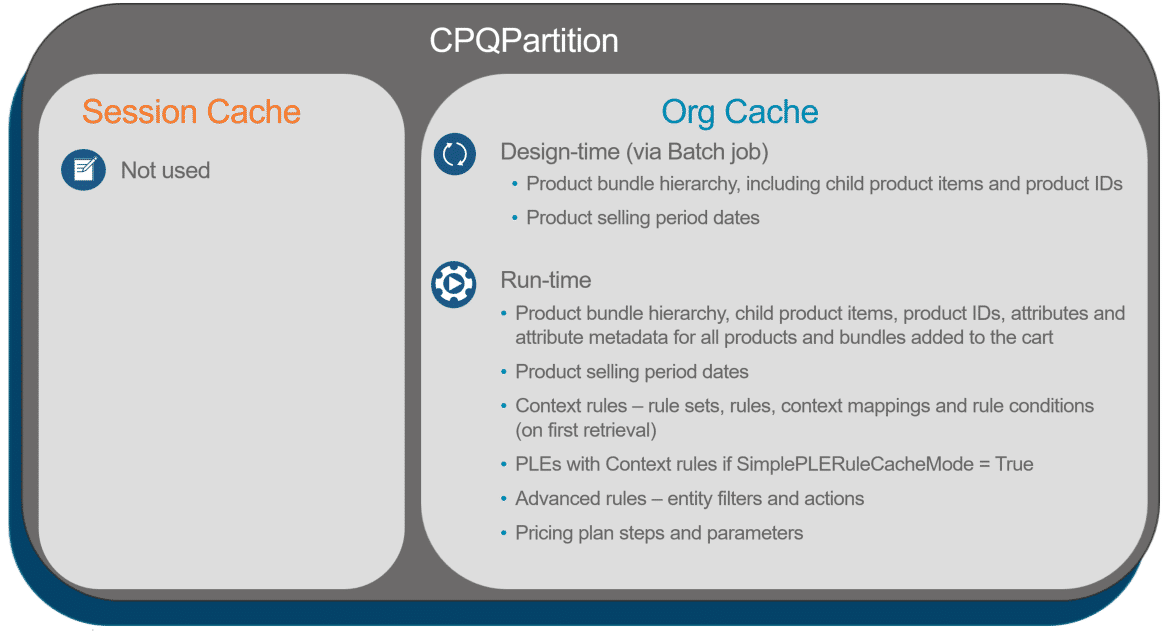
**Caching in Industries CPQ**

Caching of data is primarily a performance optimization technique for situations where the same data will be required across multiple transactions. Rather than recalculating or re-retrieving the information, data can be stored in an easy to access place and reused in later transactions.

Industries CPQ primarily caches data in a Salesforce platform cache called the “CPQPartition.” The CPQPartition is created when your managed package is installed, but you will need to enable it and ensure that it has enough space allocated. You can allocate a minimum of 10 MB for the platform cache or a maximum of whatever your Salesforce platform cache limit is. The CPQPartition, just like all platform caches, is made up of a session cache and an org cache.

You need to cache items for use by any user across the application. In the CPQPartition, the **org cache** performs this function by storing product hierarchy properties (JSON key-value pairs), rules, pricing and line item properties for your entire organization. For example, when a user creates a product bundle containing child products, the org caches the product hierarchy. So, when any user adds the product bundle to the Cart, the Cart actually retrieves the product hierarchy data from the cache.

To assist you with identifying when and where Industries CPQ transactions are saved to the org cache, refer to the graphic below.



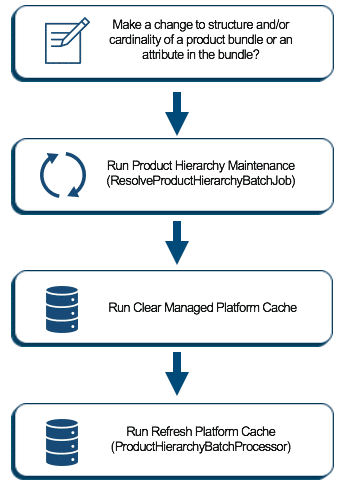
**Managing the Platform Cache**

When you need to initiate or refresh the platform cache, you use a three-step process.

1. Run the Product Hierarchy Maintenance job.
2. Run the Clear Managed Platform Cache job.
3. Run the Refresh Platform Cache job.

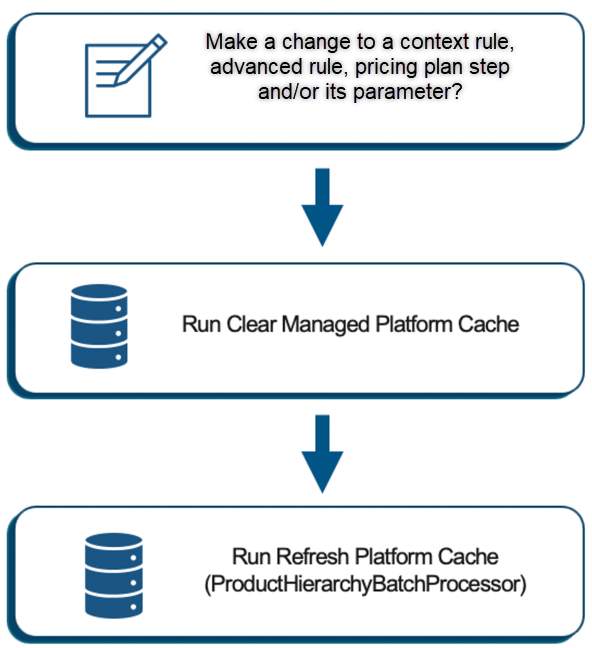
The Product Hierarchy Maintenance job builds a streamlined version of the product hierarchies in the data store sObject. Separately but related, Refresh Platform Cache invokes an Apex job that uses the product hierarchy data from the data store to build a cached version of the product hierarchy using key-value pairs.

**When do I need to run these jobs?**



Any time you make changes to product bundles (i.e., product cardinality), you must rebuild the product hierarchy data store and cache, which means you must run the Product Hierarchy Maintenance job, followed by the Clear Managed Platform Cache job and the Refresh Platform Cache job. Otherwise, you will not see your cardinality changes applied at run-time in the Cart.

You probably noticed the Clear Managed Platform Cache job that sits between the Product Hierarchy Maintenance and Refresh Platform Cache. The Clear Managed Platform Cache job deletes all data in the org cache in the platform cache. This creates a “fresh slate” for the Refresh Platform Cache job to deliver its revised product hierarchy data to the org cache. Context rules data will be repopulated when the rules are invoked at run-time. Hang tight; we'll get to rules next.



Any time you make changes to a rule (either a context rule or an advanced rule), a pricing plan step (and/or its parameter), you must clear the platform cache in order to delete the previous rules or pricing plan steps that were cached at run-time. And then you must run Refresh Platform Cache, even though you didn't make any changes to a product bundle, because the product hierarchy cache must be restored. Otherwise, you will not see the changes applied at run-time in the Cart.

When making changes to rules or pricing plans, it's not necessary to run the Product Hierarchy Maintenance job. Why? Simple, you don't need to rebuild the product hierarchy data store.

However, if you make a change to a product bundle and a rule/pricing step, follow the process flow above to rebuild the product hierarchy data store and cache.