**Communications Cloud — Modernizing the subscriber experience for both businesses and consumers.**

Communication Cloud (Comms Cloud) helps project transformation―the tipping point for transitioning to a customer-focused organization goes, by providing hundreds of pre-built, best-practice industry processes, product models, and integrations.

* **Communication Service Providers (CSPs)**have pivoted to digital and mobile channels to serve their customers.
* Comms cloud relies on cloud-based applications to automate quoting, contracts, ordering, and fulfillment of new services

Traditional manual methods such as filling out spreadsheets to track pricing, quoting, and products and a manual approval process for customer discounts are time-consuming. Comms cloud is the solution that helps the sales team manage the sales process and close the deal faster. This is when the **Configure, Price, Quote (CPQ)** platform becomes a major factor.

CPQ is a unified solution for managing a CSP’s products, pricing, rental charges, bundles, and network services.

Some of its primary functions are to:

* Configure and manage CPQ in one place.
* Interfaces and implementation framework for triggering rules.
* Guided selling experience for users to build an order.

**CPQ starts where customer relationship management (CRM) ends.**

CPQ is a technical platform that manages products and simplifies the sales process. Comms cloud is built in close conformance with common industry standards proclaimed by organizations like **TeleManagement Forum, which is often referred to as “TMF” or “TMForum.”**It provides a powerful solution for product management, product pricing, price-book, quotation, documentation, and other things you do while selling your products and services.

**What problem does this solve?**

When a company offers a wide range of products and services, managing them and maintaining their prices becomes cumbersome.

There's a lot to consider when using any manual sales process, such as

* Selling products depending on customers
* Creating accurate quotes
* Providing appropriate pricing
* Keeping track of what is done and what is not
* Providing discounts to valuable customers
* Creating promotional offers and providing them to eligible customers, etc

Comms cloud addresses these challenges and is a solid solution that streamlines your sales process and helps you create a better customer experience. Comms cloud integrates an **enterprise product catalog (EPC)**, CPQ, order management (OM), digital commerce, and pre-built processes and product models.

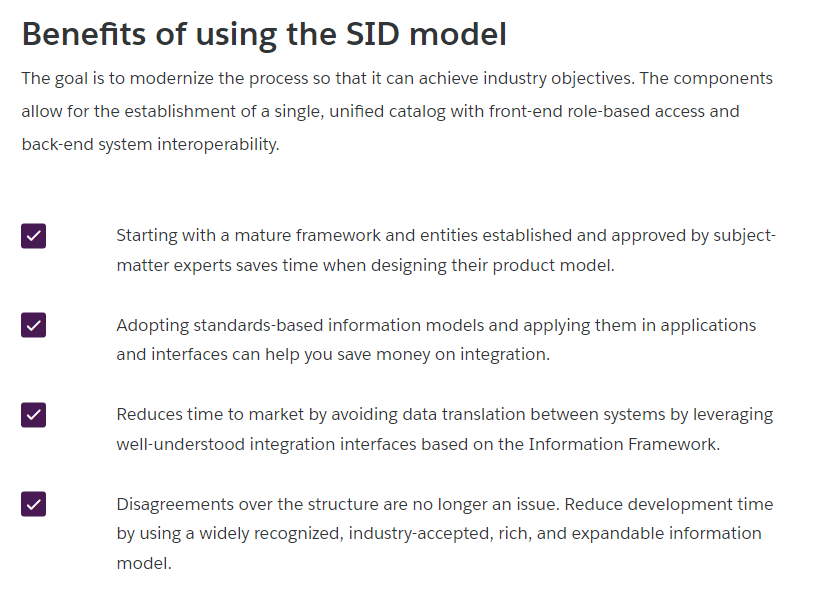
**TM Forum Compliance  
Did you know?**

**TMForum developed the Shared Information and Data (SID) framework, which established product modeling regulations.**

TMForum is a global industry association for service providers and their suppliers in the telecommunications industry. (TMForum.org)

**Product Modeling's Basic Concept**  
Compliance requires telecom business policies, customer assurances policies, and business growth policies to be evaluated through specific business entities, change-management, and implementation approaches mandated by TMForum's Information and Data Framework (SID).

* **Shared Information/Data Model (SID) model:**This modelisn’t fixed. It’s shaped based on the needs of the industry. It represents an enterprise-wide information decomposition model independent of the platform, language, and protocol.  It is a key component; it establishes standard definitions for all information that passes through the enterprise and between service providers and their business partners. The catalog structure is represented by the SID within Salesforce Industries.
* **Enhanced Telecom Operations Map (eTOM):** **The Business Process Framework** is a CSP's operating model. The model identifies important parts and how they should interact, as well as the service providers' required business procedures.
* **TAM: The Application Framework (TAM)**is a subcomponent of the TM Forum's Open Digital Framework, which provides a standard language and identifier for buyers and suppliers across all industries.



# Enterprise Product Catalog (EPC)

Some of EPC's primary functions are to:

* Introduction and management of Offers, Products, and Service configurations
* Managing Pricing configurations
* Managing business and configuration rules
* Empower catalog-driven selling and fulfillment processes providing a consistent user experience across CPQ, OM, and CLM

There are many factors in modeling and implementing a product catalog. Modeling an EPC using industry standards stresses having an overall plan for the catalog’s design. Often a CSP’s introduction of an EPC is done during a digital transformation, the launch of digital technology into all business areas, fundamentally changing how the CSP operates and delivers value to customers.

With EPC, you can:

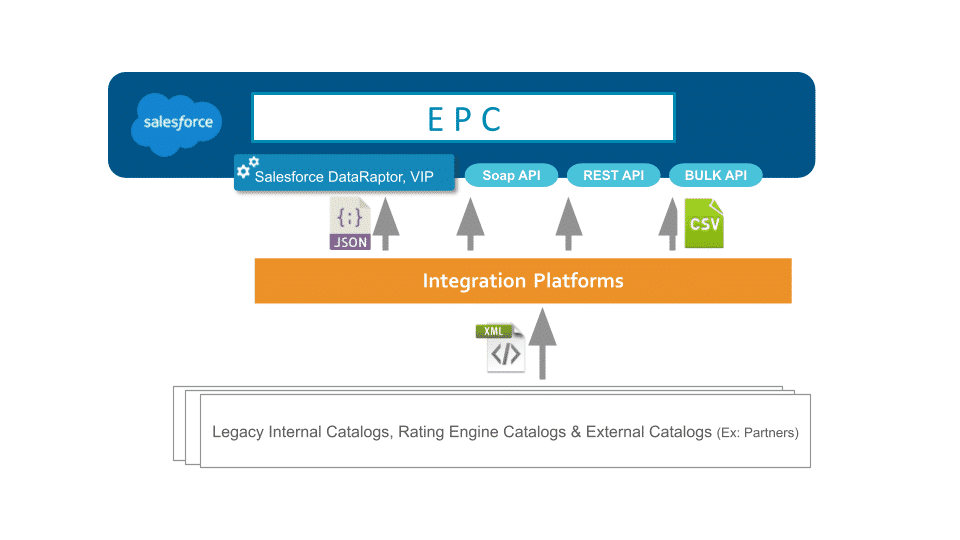
* Rapidly launch new offers and services
* Improve product lifecycle agility
* Coordinate across business function

## **Synchronizing Products**

Often you will need to synchronize products from another catalog into Comms Cloud EPC, and/or from Comms Cloud EPC into another catalog. Synchronizing products is accomplished by various means.

**Method 1**

## **Extract-Transform-Load (ETL) tool**

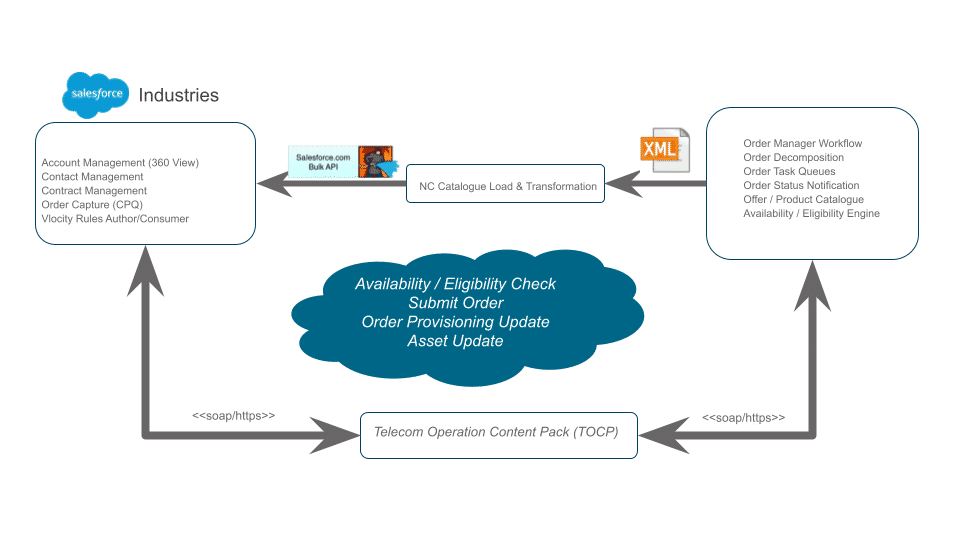


Synchronize products by using an **ETL**tool.

1. Create a payload including product information, and use the GET method that allows extracting product data from EPC only.
   1. The GET method refers to a HyperText Transfer Protocol (HTTP) method that is applied while requesting information from a particular source. It is also used to get a specific variable derived from a group
      * Future roadmap item. TM Forum (TMF620 POST) API.
      * Use the Salesforce DataRaptor and Integration Procedure
2. Make goods and other entities in Comms Cloud EPC, then use the Enterprise Product Catalog (EPC) APIs.
   1. Web APIs for import and export to partner organizations

**Method 2**

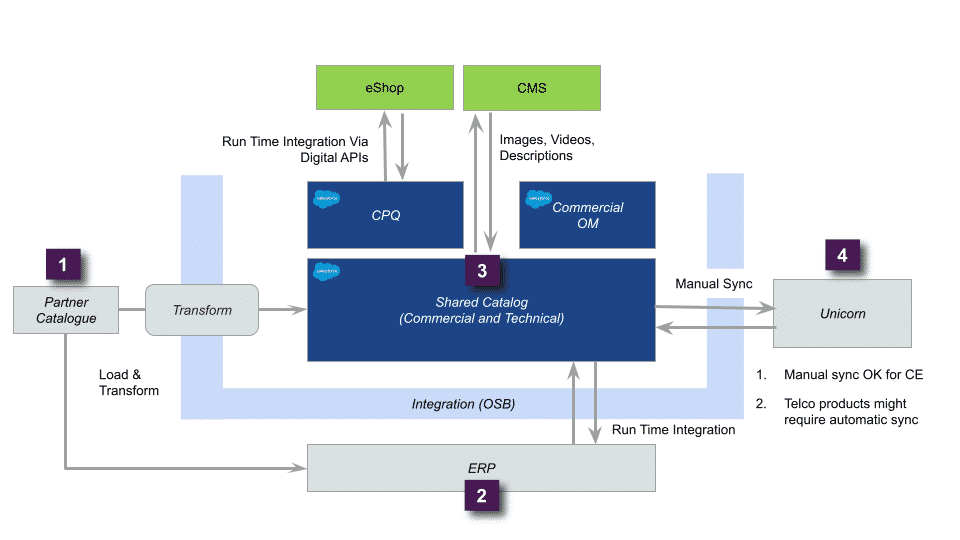
## **Salesforce Industries Flow and Integration Platform**



Use any Salesforce API: SOAP, REST, Bulk with an integration platform, allowing your product manager to have no disruption to their procedures while benefiting from the efficiencies Comms Cloud provides to the development team.  
A Salesforce industries flow accepts data and performs actions in your Salesforce org or in an external system. Screen flows are a design strategy for creating one-of-a-kind displays that collect data from a user at the same time.

**Method 3**

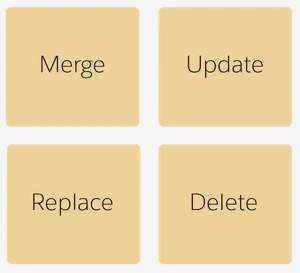
## **Catalogue Synchronization**



1. Partner Catalogue-Updates are published into Salesforce using a data transformation process that makes use of accessible platform technologies such as Salesforce (SF) bulk APIs, SOAP, and REST.
2. The company stores Enterprise Resource Planning (ERP)-related data and then becomes the cost-price king.
3. Enterprise Product Catalog (EPC) is a master for commercial product specs, pricing and rules, technical specs, decomposition rules, and orchestration plans in Salesforce Shared Catalog.
4. Manual synchronization is acceptable for Open Source (CE), while automatic synchronization is required for business items.

**Method 4**

## **Catalog Import**



**Modes**

While schemas define data file structure rules, modes define what happens to the data at import. These are the modes.

|  |  |
| --- | --- |
| With this mode... | This happens... |
| Merge | If it doesn't exist, the business manager adds new data if it doesn’t exist and updates existing data. |
| Update | The business manager updates existing data but does not add new data. It updates only the attributes that are in the XML file. |
| Replace | The business manager re-creates existing data or adds new data per the data in the XML file.  The business manager removes existing attributes that are not in the XML file. This mode is the same as a delete followed by a merge. |
| Delete | The business manager removes data in the XML file from the database. You only need to provide the object ID in the XML file.  The business manager ignores the other object attributes. |

## **Summary**

The solution will differ significantly based on the requirements and specific systems/vendors involved. Some customers may not do an automated catalog synchronization at all or may opt to do one at a later stage of the program.

# Catalog-Driven Approach

**The catalog-driven approach means there is no code, processes, or templates. Specific offers or promotions are dynamically referenced in a catalog.**

**For Example**

**Universal Cart**

* Using a product-catalog-driven approach means the system works based on what is defined in the catalog.

**Customer service providers (CSPs)** equipped with commerce capabilities provide the best experience for the customer: where they access a single, universal cart through any channel.

**Order Management**

* The catalog-driven architecture separates all aspects of product representation as a configuration activity driven by data, not code.

This solution simplifies the role of the product manager using a catalog-driven solution with versioning and templates, which use an easy-to-manage **configure-price-quote (CPQ)** tool and seamlessly flow into a powerful, dynamic commercial order management system. Customers then see a standard cart as they switch channels. Customer-facing personnel in retail stores and the contact center see a complete view of abandoned carts, carts in progress, and recently completed carts.

**Digital Marketing**

* The primary driver is to allow commercial product representations to be changed quickly and outside typical IT delivery timelines.

CSPs nurture the customer through a digital marketing journey. Product managers avoid duplicate entries and respond quickly as new products and services enter the market – or competitors change the game. Customers are retained — and upsold — with AI-powered personalized recommendations and promotions.

**Initial Drivers**

When a communication service provider (CSP) decides to adopt a catalog-driven approach, one of the main drivers is meeting the major differences between marketing teams and fulfillment or technical organizations looking at service. It allows a separate but unified experience for each marketing channel. This is achieved by breaking down services into self-contained, atomic reusable building blocks within the catalog and applying the SID model.

Technical team members focus on the complex technology required to deliver and retain services while providing effective tools for marketing to create and roll out new offers, discounts, campaigns, and promotions to the market.

**Benefits of Adoption**

CSPs can use a central-catalog-driven architecture to become flexible digital service providers with omnichannel capabilities, launching new services, bundles, and offers fast and generating additional income from third-party services.

* **Streamlining**: The catalog approach forces a CSP to revisit and evaluate all of its product offerings and remove any that aren’t worth supporting. The process may be laborious, but the ultimate benefits are immeasurable and absolutely worth it for the benefit of the entire business.
* **Normalization**: Adopting the standard information and data model (SID) forces the CSP to normalize its product definitions. This pays dividends in integration, data migration, data cleansing, and internal data standardization. These all have real, measurable costs and long-term management benefits that transcend the product catalog.
* **Personalization:** With atomic building blocks, normalized product definitions, and spring cleaning of the product set completed, the CSP creates streamlined sales processes to bring those products to market, introducing personalization and bridging the relationships between product characteristics and personalization triggers which results in predictable outcomes.
* **Automation**: The enterprise product catalog provides a key and central trigger point for automation service fulfillment. It enables the CSP to abstract complex business, network, and design rules – as well as enable one-touch provisioning starting from the product catalog, but also to replace, redesign, and improve the underlying fulfillment systems without impacting the catalog itself and the customer-facing processes it supports and enables.

**Consequences if you DON'T use a catalog-driven approach?**

If you don’t use the catalog-driven approach, **data is pulled from hard code prices, product names, etc.** It’s difficult to maintain and make changes.  
Not using the catalog-driven approach could also cause **order fallout**. This is when an order fails, for example, due to a mismatch between the product defined in Comms Cloud and the product in the backend.

**Benefits if you use the catalog-driven approach?**

* Using this approach, your **data is organized** and easy to manage.
* Designing with a product-catalog-driven approach **aligns services and products, clarifies concepts, and reviews existing input**, which offers result details within the processes.

This approach also allows the project team to map and see all parts of your business and make product models available to everyone.

# Commercial and Technical Catalog

Product modeling is approached by separating two aspects of the product to distinguish between **commercial** and **technical**catalog items. This distinction derives alogical boundary within the catalog:

* A **commercial**element of the catalog consists of all the commercial products sellable to customers.
* The **technical**element of the catalog consists of the products that hold the service layer information for the corresponding commercial products, and these technical products are non-orderable and not visible in the cart. Technical products are used to provision the product and services sold to the customers

**Consideration**

Why is it important to differentiate between commercial and technical catalog product elements?

* To avoid loading unnecessary technical information into the **CPQ (Configure, Price, Quote)** cart, which is not required by the customer.
* To avoid creating service layer products under the bundle or individual offers, just pass the information to the provisioning system.
* To avoid the deep hierarchies of bundles.

You model the Commercial Catalog from the perspective of the order capture and assets created for a customer.

**Decomposition**

Decomposition falls into the Order Management (OM) space. For example, your fulfillment systems have a complex legacy model from a technical backend perspective. The fact that the model’s complex doesn’t mean that the way you sell it needs to be complex from a commercial perspective. You can create a technical product to share with all products of the same class. Therefore, define a single decomposition relationship once for the class instead of for each commercial product.

To share decomposition configurations across multiple products:

* Create a product with a record type “Class” and define the decomposition against this product.
* Create new offers that reference this new class by populating the Parent Class field with the product you created in step one.

Model the technical catalog to meet the technical requirements of fulfillment and to make the integration with fulfillment systems easier.

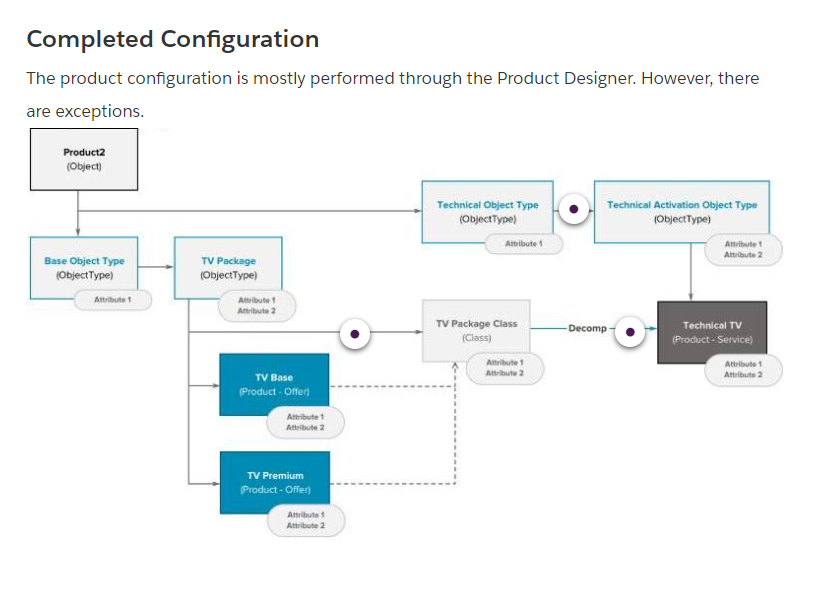
A fulfillment designer must create corresponding decomposition rules to process orders containing commercial products. This may result in a very large technical catalog and many repetitive, potentially error-prone rules that are difficult to maintain.

**Parent Class to Offer Relationships**

You can use ObjectType with commercial and technical products for more effective modeling. To enable class-based decomposition, the fulfillment designer creates a class and defines the decomposition rules against the class, then updates the offers to set the Parent Class field to point to the class.

Catalog designers can use ObjectType to model products that share the same traits. Another example is a variety of commercial broadband offerings, such as bronze, silver, and gold, that are all realized through the same Broadband fulfillment process. For example, a TV service may have many channels or channel packages that can be defined that all share the same fulfillment process.

OM doesn't currently use ObjectTypes but does use the parent class concept. Catalog designers can take advantage of both concepts simultaneously to optimize how decomposition works.

Since the decomposition function maps attributes and fields to technical products, the class must include the same attributes and fields in its configuration. This can greatly reduce the amount of work and better control attribute definitions.

# Product Modeling in EPC

**There are some guidelines that you use for product modeling; however, a lot of it is based on experience, project**

**Considerations**

In catalog modeling, it's difficult to define a rigid and fixed model to apply in every context. It is strongly influenced by the customer requirements, the business context, and some highly volatile requirements such as pricing rules, eligibility rules, and provisioning rules.  Sometimes, depending on the country, region, or company, even the same business scenario there might have a set of diverging requirements:

* Pricing logic
* Eligibility
* Provisioning
* Requested UI/UX

The EPC architecture supports the product modeling development process. From a high-level perspective, the catalog-driven approach can be seen as a three-step iterative process:

1. **Requirement collection**: Before defining any kind of modeling, the EPC architect needs to collect the list of minimum information requested (aka discovery) to decide how to design the requirement properly.
2. **Application of modeling pattern:** Apply the correct modeling patterns to represent and model the information collected during the requirement collection step.
3. **Model approval:**Evaluate the overall modeling resulting from the previous steps against the main functional and performance requirements.

**Requirement Collection**

The first step to catalog modeling is to collect all the business, functional, and technical requirements for a proper design. Typically business needs are represented in the EPC with multiple approaches based on the collected information. So, it’s important to have a full and correct understanding of the requirements to avoid taking the wrong direction during the design process.

Requirement collection is vital to selecting the correct modeling pattern and applying the model.

For instance, a business narrows its focus to a limited set of information and only answers discovery questions with priority to that particular area.

**Application of Modeling Pattern**

**What is the correct modeling pattern?**

While designing an application, architects often stumble on an issue where they need to model their data catering to the organization's structure, the product, and the consumer's or clients' wants. It's a difficult charge to consider all of the components.

**Model Approval**

**Not Equal, But Balanced**

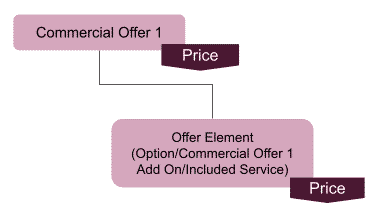
Equal doesn't mean balance. You can't satisfy everyone when architecting a solution. Often the right approach means it isn't exactly what the business wants but is the best approach.

# Modeling Patterns

**A requirement is often modeled with different strategies during the catalog design process.**

**Hierarchical**

Catalog items are elements of a single offer related by a hierarchical relationship to reflect the collected business and technical requirements.



**A hierarchical modeling pattern is a common pattern used in product modeling.**

The hierarchical pattern efficiently manages typical commercial requirements such as product and group cardinality, hierarchical representation of products, and product dependencies.

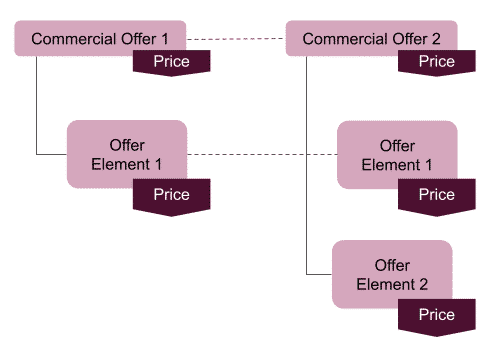
It’s usually applied to model different offers and products with solid dependency in terms of:

* The hierarchical relationship required by a business or technical constraints
* Cardinality rules
* Required and excluded rules

The following table summarizes the pros and cons of using the hierarchical pattern

|  |  |
| --- | --- |
| Pattern Description | A hierarchical pattern is a design option when the following conditions are met:   * Bundled items are an offer option or add-on not sold as a standalone offer. * Bundled items have an explicit price. * A request for a hierarchical representation of the item composing the commercial offer is made. * Different commercial offers have the same service with different cardinality and prices. * It contains both a single relationship and group cardinality that applies to embedded bundles within a hierarchy. * The number of items in the bundle increases. A linear representation might have these negative results:   + Complex to read for the end-user   + Hard to configure for the administrator by forcing a hover of advanced rules, and complex Move, Add, Change, Delete (MACD) logic |
| Pros and Cons | The following bullet points summarize the pros and cons of the hierarchical pattern:  **Pros**   * Efficiently manage the cardinality requirements without using advanced rules, such as the minimum/maximum/default cardinality and group cardinality. * Define reusable components across different EPC bundles, saving time and offloading the maintenance effort. * Cardinality requirements of product bundle components are easily satisfied in a hierarchical product structure   **Cons**   * The front-end application of order capture needs to render the complete product hierarchy of mandatory as well as optional add-on children products. |
| Consideration | In the telecommunications industry, hierarchical modeling using product bundles is the most commonly used pattern that's implemented across the traditional IT systems used by Communications Service Providers (CSPs). The hierarchical model is a bundling system that supports several types like static bundles, and configurable bundles, kits, and feature choices. These functions are commonly used in salesforce as well as other sectors.  The existing product design guideline is influenced by the current capabilities of CPQ, which are considered to be relatively limited. Linear and hierarchical product designs do not have to be mutually exclusive; they can coexist.  For example, there can be linear interactions between two hierarchical systems.   * Don’t use hierarchical representation to represent the relationship between the services and the infrastructure (devices and hardware) that must be accessed. * The hierarchical structure should omit any product used to represent service fees, such as shipping fees, installation fees, or penalties. * Instead of using a hierarchical representation, use the Catalog Category entity to represent catalog categories. |

**Offer Meiosis**



**An architect defines the different items as separate, independent products.**

The following table summarizes the pros and cons of using the offer meiosis pattern

|  |  |
| --- | --- |
| Pattern Description | An offer meiosis pattern is a design option when the following conditions are met:   * A bundle add-on defines a different offer than the initial bundle from a business perspective. * Bundle items have an explicit price. * A hierarchical representation of the item composing the commercial offer. * The same service belongs to different commercial offers with different cardinality and prices. * There is a wide need for cardinality rules in a single relationship and group cardinality. * The number of items within the bundle starts to increase. A linear representation might be too complex for the end-user to read. * The two offers have some diverging configurations, such as these examples:   + Offer A includes some options not included in offer B.   + Offer A and B have two different prices for the same service. |
| Pros and Cons | The following bullet points summarize the pros and cons of the hierarchical pattern:  **Pros**   * It’s the best available pattern to represent the availability of multiple exclusive offers with diverging configurations. * It provides an accurate and detailed representation of the actions and changes performed by the MACD process while moving from a source offer to a target one. * Every single item is properly marked:   + Newly added items   + Removed item since it’s not compatible with the target offer   + Item mapped from one offer to another   This provides excellent advantages:   * Order management provides the complete list of actions to perform during the change order. * End-user: It gives a clear functional understanding of what is kept or replaced during the MACD process. * It strongly limits the number of advanced rules, pricing, eligibility rules, and process customization needed to represent the swap between two mutually exclusive offers.   **Cons**   * Large structures may affect the getCartItems performance. * Large hierarchical structures may affect general Configure, Price, Quote (CPQ) application programming interface (API) performance. * The front-end application manages the getOffer details by displaying the option and add-ons part of the hierarchy. * The Change of Plan capability tends to create orders with many line items. * The different offers do not support the inheritance of promotions and discounts during the Change of Plan process. The business must consider them as separate commercial offers. * If used to move across two large bundles, the Change of Plan capability might generate orders with many line items. |
| Consideration | The offer meiosis could mix the linear and hierarchical patterns. This mix is a different pattern because it represents a different and distinct modeling option compared to the linear, hierarchical, and attribute-based patterns.   * Carefully evaluate how the model is applied and used during the MACD scenarios. * EPC architects should carefully evaluate how this model is applied and used during the MACD scenarios to avoid creating bulky orders. * Hierarchical representation doesn’t represent catalog categories. * Hierarchical structures should omit any product used to represent service fees, such as shipping fees, installation fees, and penalties.   Let’s walk through a business case scenario to understand this choice better.    Given a standard entertainment commercial offer called “Entertainment,” a business wants to add an optional package with premium entertainment channels for a special additional price. |

**Linear / Relies-On**



**An architect defines the different items as separate, independent products.**

Whether the initial order capture is easy in a CPQ journey. It has a downstream impact on the order submits payload and considerations for asset-based ordering. In addition, the relies-on approach isn't fully supported out of the box, isn't a productized feature, and requires customization.

**Product order comprises four order lines (orderItems).**

**Product order composes of four order lines (orderItems)**

**Line 1:** Purchasing of a new simple product that needs a physical delivery place and an appointment to be delivered

**Line 2:** Modification of a characteristic value of an already owned product and changing the user associated with this product

**Line 3:** Purchasing of a new simple product that needs (is supported by) another already owned product (in this case, for example, a new product that **relies on** the owned mobile access product to be installed)

**Line 4:** Purchasing a bundled product composed of two atomic products

The linear pattern provides a general performance boost in many business scenarios and generates an order with a low number of XLI ( Order, Quote, Asset Line Item), reducing the  Data Manipulation Language (DML) impact on the application performance.  
  
Use linear when you have to model different offers and products without any strong dependency in terms of:

* Hierarchical relationship required by the business or technical constraints
* Cardinality rules
* Requires/exclude rules

The following table summarizes the pros and cons of using the linear pattern

|  |  |
| --- | --- |
| Pattern Description | Every creative endeavor requires that you take risks. If you try and don't succeed, you've still learned something. It took Thomas Edison more than 10,000 tries to invent a usable lightbulb. You're not failing. You're discovering what doesn't work. |
| Pros and Cons | A linear pattern is an option when the following conditions are met:   * The new item is not requested to be represented as hierarchically related to another commercial offer. * The item is not requested to have different price or cardinality rules based on other commercial offers. * The number of offers and relationships managed with advanced rules is limited, and the advanced rules don’t have complex conditions. |
| Consideration | For limited cardinality rules or product relationships, the same model may be applied by relying on some product capability, such as the following:   * Advanced Rules: Auto Add, Auto Remove, Requires, Exclude * Product relationship: Such as Requires, Relies On, Excludes, and Recommends   This mainly applies to Offering Management (OM)-specific requirements with a request to represent a connection between two separate commercial offers that have provisioning dependencies or connections.  For example, the Sports entertainment package is a standalone commercial offer that can be sold independently from any other offer, but it relies on a specific setup box or smartcard. |

**Attributed-Based**



**This pattern represents different products, services, and product  
characteristics as a product attribute or attribute value.**

The following table summarizes the pros and cons of incorporating attributes in your modeling pattern.

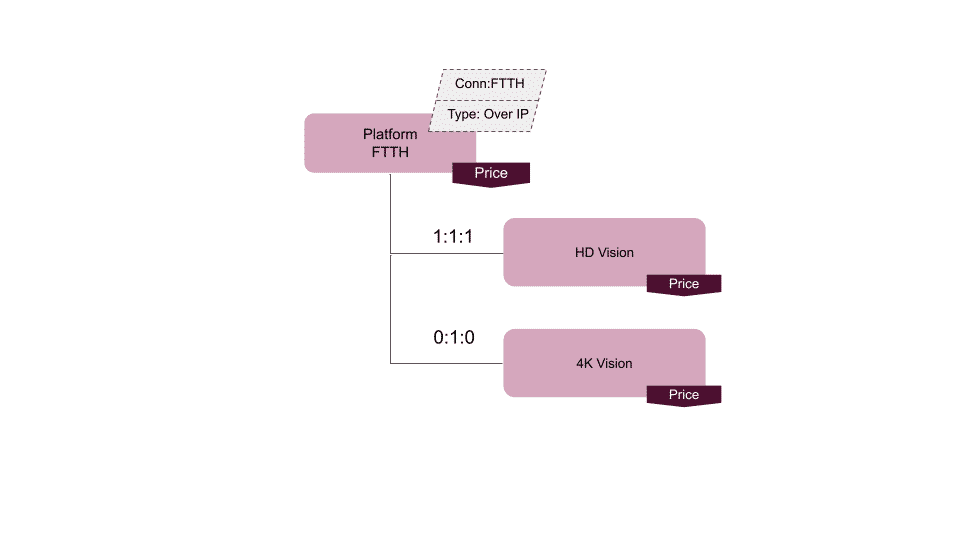
|  |  |
| --- | --- |
| Pattern Description | An attribute-based pattern is a design option for items that meet the following criteria:   * The item is seen as a characteristic of the main offer and is not a standalone offer. * The item does not have an explicit price. * The item is not subject to the target of advanced or eligibility rules. * From a functional perspective, the item should be a characteristic of the assigned product. |
| Pros and Cons | **Pros**   * It reduces the number of order items to be managed during sales and post-sales orders, leading to better application performances for any post or operation. * It strongly reduces the payload size and complexity used invoking the CPQ API. It represents the best way to model service or product characteristics.   **Cons**   * Eligibility rules based on attributes might be complex to configure. * Attributes are generally more complex to read and change from a Digital Commerce and Cart-based API perspective. * Attributes do not support both eligibility and advanced rules such as these product relationship types:   + Auto-Add   + Auto Remove   + Required   + Excludes |
| Consideration | EPC is where you model and define the technical products to use in the decomposition and fulfillment steps. EPC provides the concept of ObjectType to confer the aspect of inheritance to the product model.  Modeling the Commercial Catalog using Object Types is standard across all projects. However, the same feature also can be useful for technical products.  Specifications in the Technical Catalog are defined like other products, meaning that most of the fields you configure for the Commercial Catalog also apply. However, there are some differences: The technical specifications do not need pricing, and the product should never be orderable. There are likely no attachments as the product is never exposed in CPQ.  Order Management only uses some configurations, such as the Scope field. The Scope field is critical in controlling the decomposition behavior.    You also can manage product attributes through EPC and use appropriately designed layouts to simplify product creation. Therefore, it’s highly recommended that you design the Technical Catalog using Object Types.    You model the Commercial Catalog from the perspective of the order capture and assets created for a customer. However, you model the Technical Catalog to meet the technical fulfillment requirements and simplify integration with fulfillment systems.  Here are the two main approaches to modeling a Technical Catalog:   * **Hierarchical Modeling:** This model comprises the Customer-Facing Service (CFS) and RFS layers. The CFS layer confers an abstraction of the RFS layer that is less technical and more understandable from the commercial side. * **Linear Modeling: This model is only concerned with the RFS layer and is modeled to support the integration with fulfillment systems. Linear modeling is typically less complex because of fewer decomposition relationships and mappings.** |

# Examples of Modeling Patterns Traditionally, a catalog is commonly the result of a recurrent application.

## Scenario One

A communications company, ACME Entertainment, aims to launch a Minimal Viable Product (MVP) as the following commercial offer.

## Mix Of Hierarchical And Attributed-Based Patterns



# The EPC architect applied the hierarchical pattern and defined a new bundle of products and services by creating a new offer called “Platform.” The attribute-based pattern defines the connection and the platform type because they represent the characteristics of the Platform product. Therefore no need to have a hierarchical relationship with the Platform product or an exact price from a business perspective.

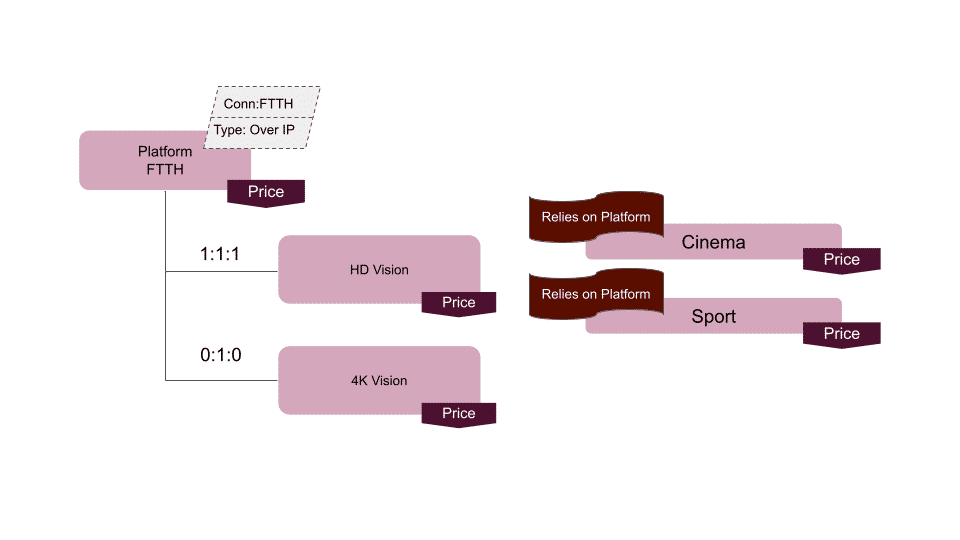
## Offer Meiosis Model Of Offers

# Offer Meiosis Scenario One

The two entertainment offers (Sport and Cinema) are independent of each other. The subscriber decides to subscribe to neither or one without any restrictions. Even from a pricing perspective, the two offers are independent. For this reason, the EPC architect uses the full-flat pattern, as shown in the image below.

The number of TV channels is high and they are frequently introduced or removed as in-market offers. Keeping them tightly coupled within the plan bundle hierarchy will lead to performance and ongoing maintenance issues. Hence an alternate approach would be to keep the channels outside the plan bundle and connect them via relies on relationships in the cart context.

## Final Model Of Offer With Relies-On Relationship



# To represent a connection between the entertainment packages and the related platform there is a hierarchal relationship between HD vision and 4K vision. The EPC architect defines a relies-on relationship between the two entertainment packages and the Platform. Note: Relies-on is not a productized feature and requires customization.

## Scenario Two

Six months after the initial launch, the ACME Entertainment wants to extend its market presence and territorial coverage by providing the same entertainment packages on satellite technology.

# With the subscription of ACME Entertainment packages on the satellite platform, a customer can subscribe to the ACM Entertainment service in areas not covered by the FTTH technology. The satellite offering includes the HD and 4K vision at a special price of 0$ and a new service name playback that allows a customer to record and store a live broadcast on its decoder to watch later.

## Modeling

The Platform Satellite is a mutually exclusive alternative to the Platform FTTH. The two platforms include some common services with different cardinality and pricing rules, such as the 4K Vision and other services or devices that are platform-specific. The EPC architect models the new requirement using the Offer Meiosis pattern for these reasons.

## Satellite and Platform Offers Model

# Satellite Offer Model

# A new bundle Platform Satellite is modeled similarly to the FTTH Platform with some differences, such as the 4K Vision pricing and cardinality plus the Playback service. The following image shows the result.

# Product Life Cycle

Product lifecycle is the length of time a product is launched to consumers into the market until it's pulled from the shelves. The lifecycle is broken into five stages: ideation, product configuration, production launch, change management and retire.

This cycle is conceptualized by architects to determine the best approach to develop their product model and the process of strategizing ways to continuously support the evolution of the product and maintain the product.

# 

# 