**Getting Started with Context Rules**

In the Order Capture process, rules are essential to creating the perfect order, giving you control of all the products and services offered to your customer, including offering them at the right prices.

You can also use rules to assess the appropriate penalties for product or service changes and cancellations in line with your business objectives.

It’s important to understand your customer requirements end-to-end before you decide on which types of rule to use.

Salesforce Industries provides two rules frameworks to accomplish your business objectives: **Context Rules** and **Advanced Rules**.

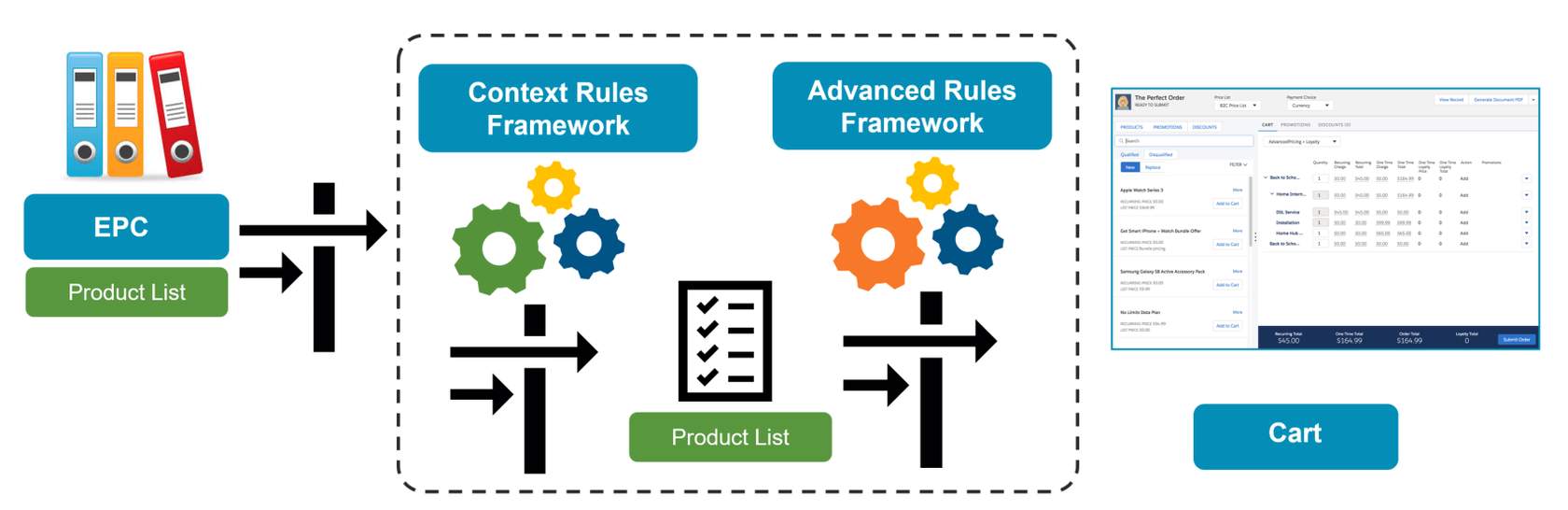
**Dual Frameworks, Working Together**

The Context Rules and Advanced Rules frameworks work in tandem to run business rules in your Industries CPQ environment. These business rules tailor the Shared Catalog (EPC) data into product and service offers that are relevant and available to customers.

**Context Rules** qualify products, promotions, price lists, price list entries and pricing adjustments in the Cart. This framework can also be used for Salesforce Industries API Caching, which is used for digital commerce.

**Advanced Rules** is Salesforce Industries' original rules framework, and it is used primarily to create rules for product compatibility or configuration. It's important to understand each framework's strengths in order to decide which type of rule to use.

Both rules frameworks operate in tandem to provide comprehensive rules to govern all Industries CPQ interactions.



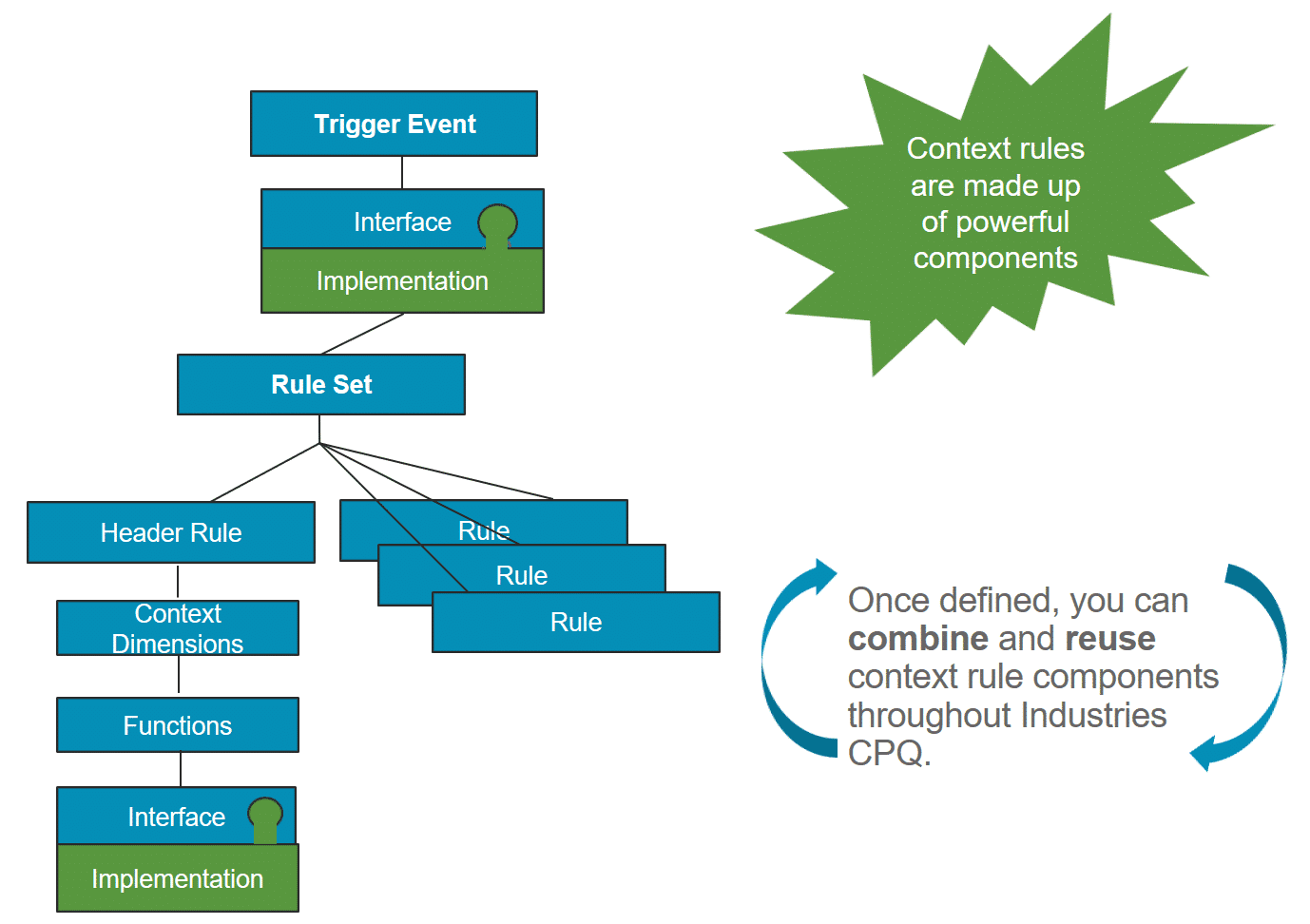
The diagram shows how the two rules frameworks work together in the Cart. The process begins by gathering all active products in EPC, moves through the Context Rules Framework to filter the product list, then to the Advanced Rules Framework to further refine the product list, and finally presents available and eligible products and promotions in the Cart.

# Context Rules Framework and Rule Types

**The Context Rules Framework**

The Context Rules Framework allows you to build context rules to apply qualification and penalty rules. The framework includes the following elements, as shown below:

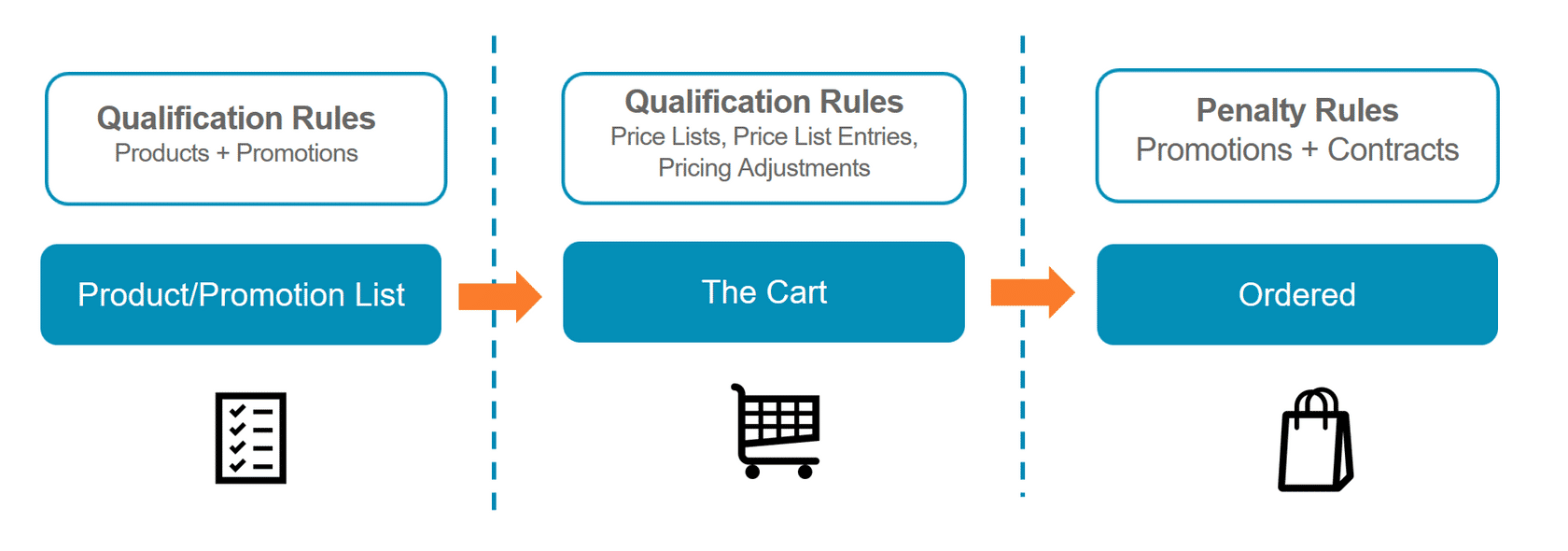
* Trigger events (such as adding a product to the Cart) that run an Industries CPQ interface, which then calls the active interface implementation
* Rule sets that run one or more context rules
* At least one context rule, which includes a context dimension, context scope and context mappings.



**Context Rule Types**

In the Product Console, you can create the following types of context rules:

* **Qualification:**This rule type determines eligible products and promotions for a customer before you add them to the Cart, and determine eligible price lists, price list entries, and pricing adjustments for products in the Cart.
* **Penalty:** This rule type determines penalties for a customer who cancels contracts or promotions they've already ordered.



**Qualification Rules**

A qualification rule determines the eligibility of a customer to receive a product or promotion before it is added to the Cart. Create these rules to filter the Products and Promotions lists in the Cart so that customers only see the products for which they're eligible. For example, a product or promotion is only available for purchase by long-term customers.

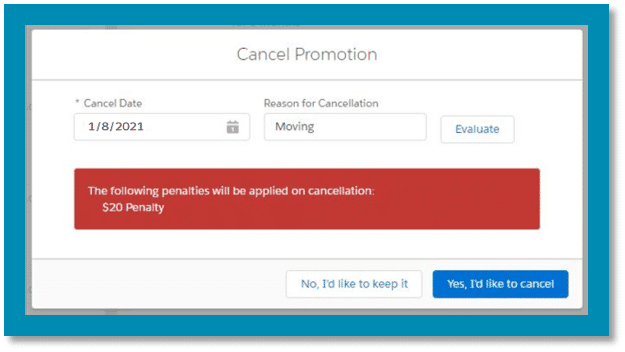
You can also use qualification rules to determine eligible price lists and price list entries for a product or promotion in the Cart. For example, a customer in CA is eligible for pricing from the West Coast price list. Qualification rules can also control what pricing adjustments can be applied and under what circumstances.

**Penalty Rules**

Penalty rules apply to promotions that have already been ordered by the customer or to contracts that are cancelled during the contract period. In the case of promotion penalty rules, if the customer cancels the promotion during the contract period, the rules determine whether a penalty applies to the customer's account. If a penalty applies, the rule defines the penalty charge (a specific fee).

Example: A service has a 12-month subscription. If the customers cancels the service before the end of 12 months, a penalty fee is applied to their account.

Penalty rules apply only to MACD or assetized orders (which we cover in the Order Capture course).

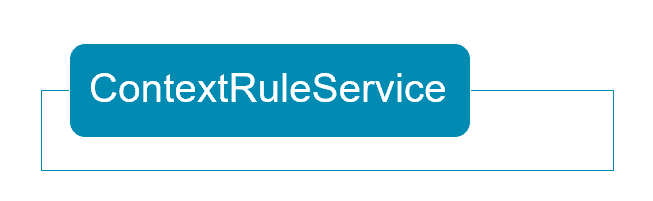


If a penalty rule is triggered, the user is prompted that a penalty will be applied if the promotion is canceled. The user can continue with the cancellation or keep the promotion.

# Interface Implementations for Context Rules

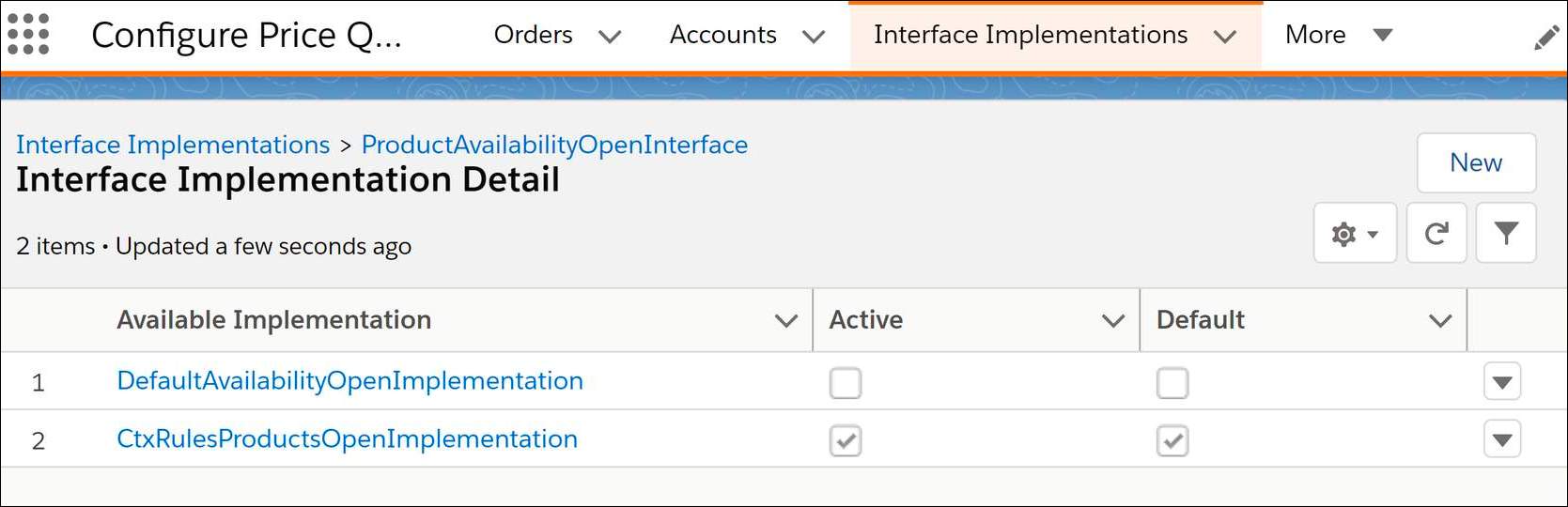
As you learn about context rules, it's important to know which interfaces and implementations are used in the context rules framework. The primary interfaces and their implementations are listed below.

**ContextRuleService Interface**



The **ContextRuleService** interface has only one implementation, which is also named ContextRuleService. Easy to remember, right? This implementation must be the active and default implementation in order for context rules to be enabled.

**Qualification Rules for Products: ProductAvailabilityOpenInterface**



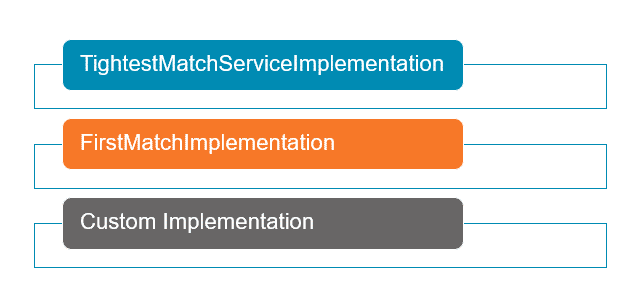
The **ProductAvailabilityOpenInterface** is called when the application must provide a list of products that are available to the customer to select. This interface has two implementations:

* The **DefaultAvailabilityOpenImplementation**returns the list of products without running context rules, so would only be used when you do not have any context rules to apply.
* The **CtxRulesProductsOpenImplementation**returns only those products which meet the context rules set. This must be the active and default implementation in order to deploy context rules for products.

It is possible for customers to have a custom implementation for this purpose if required.

There is no similar interface for context rules for promotions. It's just included as part of ContextRuleService.

**Qualification Rules for Price List Entries: TightestMatchInterface**



The **TightestMatchInterface** is called when the application must price a product in the cart.

If the **TightestMatchServiceImplementation** is active and default, this interface will use weights to determine which price list entry to select.

If **FirstMatchImplementation** is active and default, this interface ignores those weights and selects the first match that it finds.

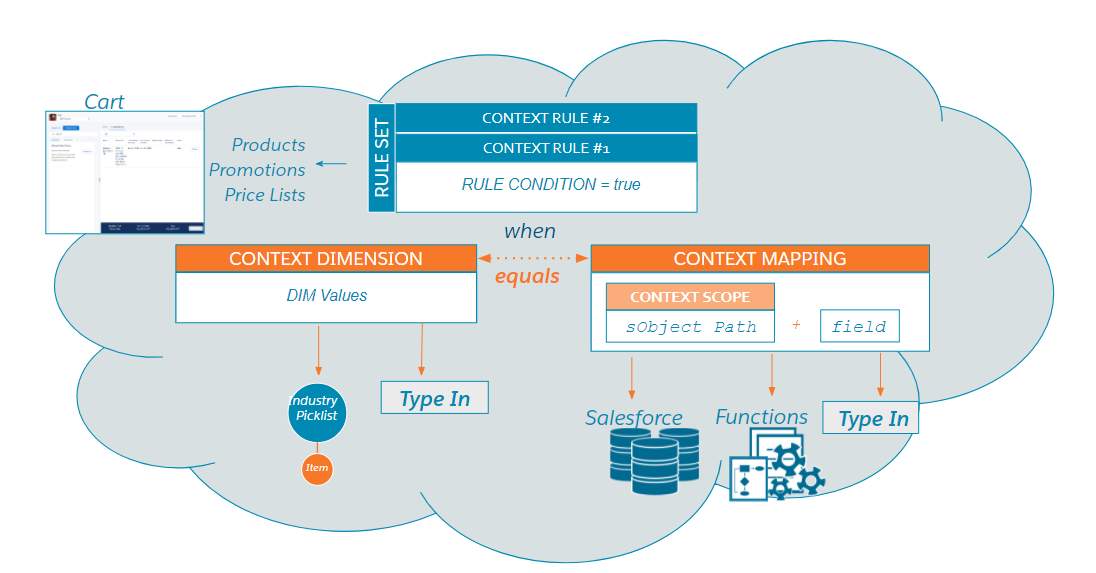
**Additional Context Rules Interfaces**

In addition the interfaces listed above, there are two other important context rule interfaces:

* **PricingManAdjEligibilityInterface**: For qualification context rules for pricing adjustments
* **RepricingManAdjEligibilityInterface**: For qualification context rules for pricing adjustments used in a repricing batch job

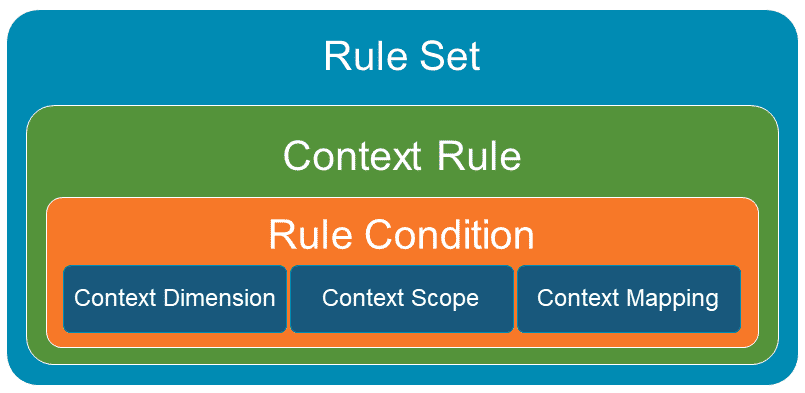
# Putting the Components Together

All the elements of context rules work together in the Context Rules Framework to filter products, promotions, price lists, and price list entries from the Shared Catalog (EPC) to the Cart.



# Rule Sets

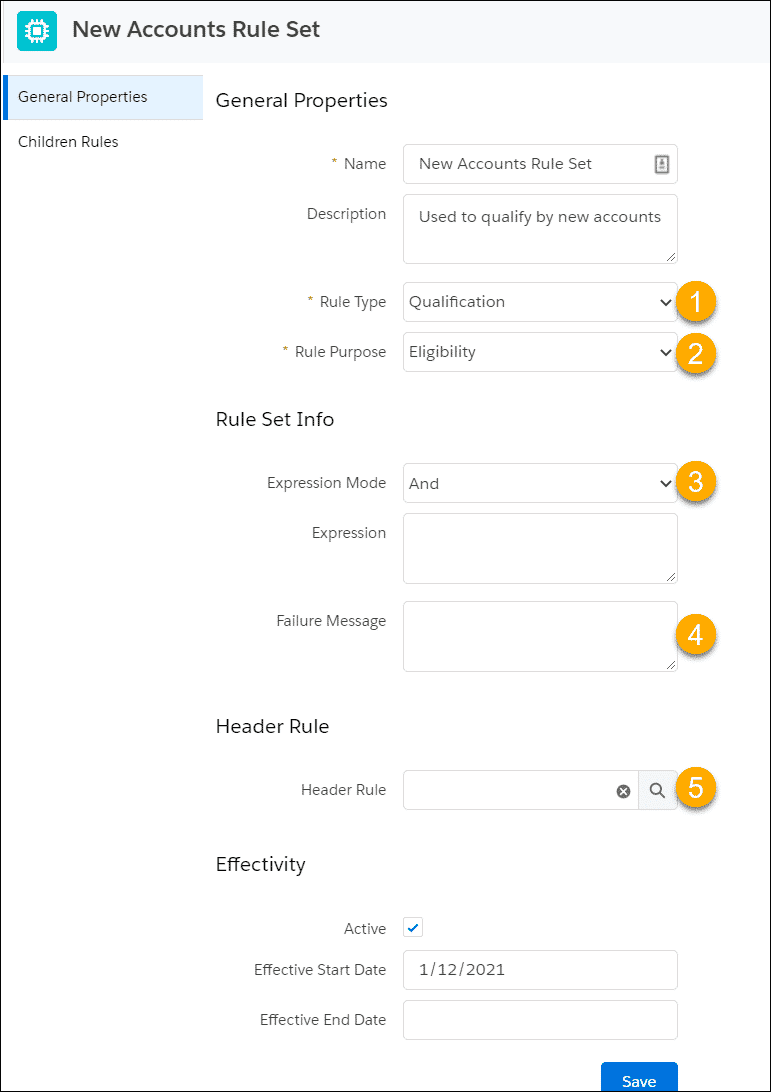
**What's a Rule Set?**



A rule set is a collection of one or more context rules applied to a product, promotion, price list, or price list entries. These rules are evaluated as a whole when performing a check against a product or promotion.

* To express complex conditions, you must group rules into a rule set.
* You apply rule sets to products or promotions, rather than individual context rules.
* Rule sets are stored as Vlocity Rule objects.

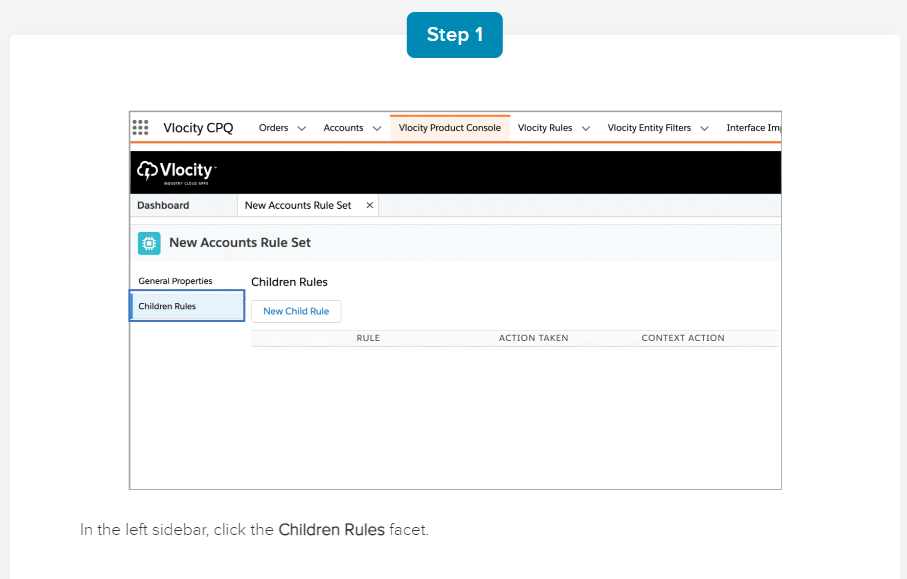
**Creating a Rule Set**

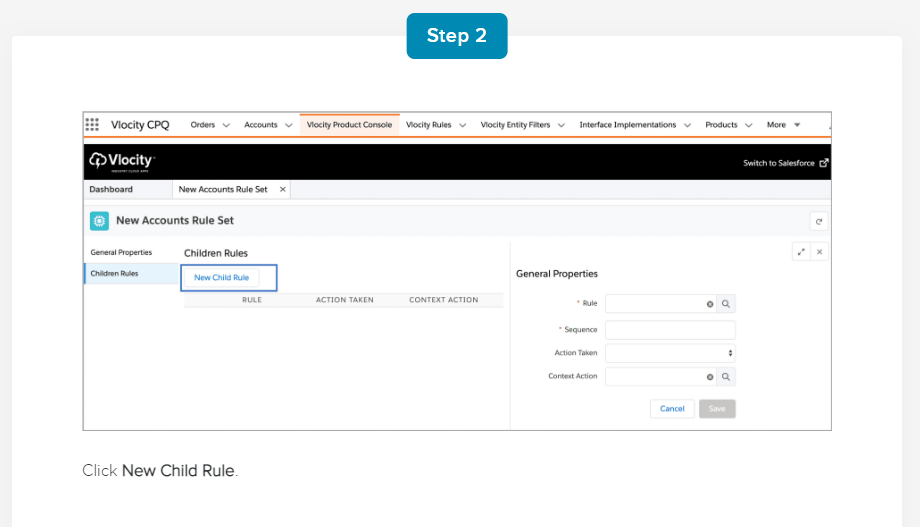


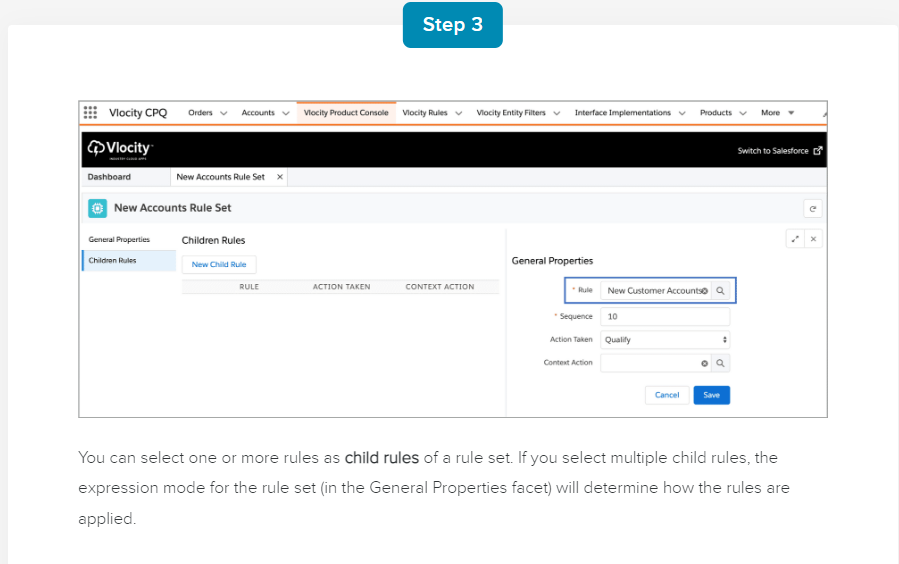
From the Vlocity Product Console Dashboard:  
Under Rules, click **+** next to **Rule Set**. Set the general properties and rule info:

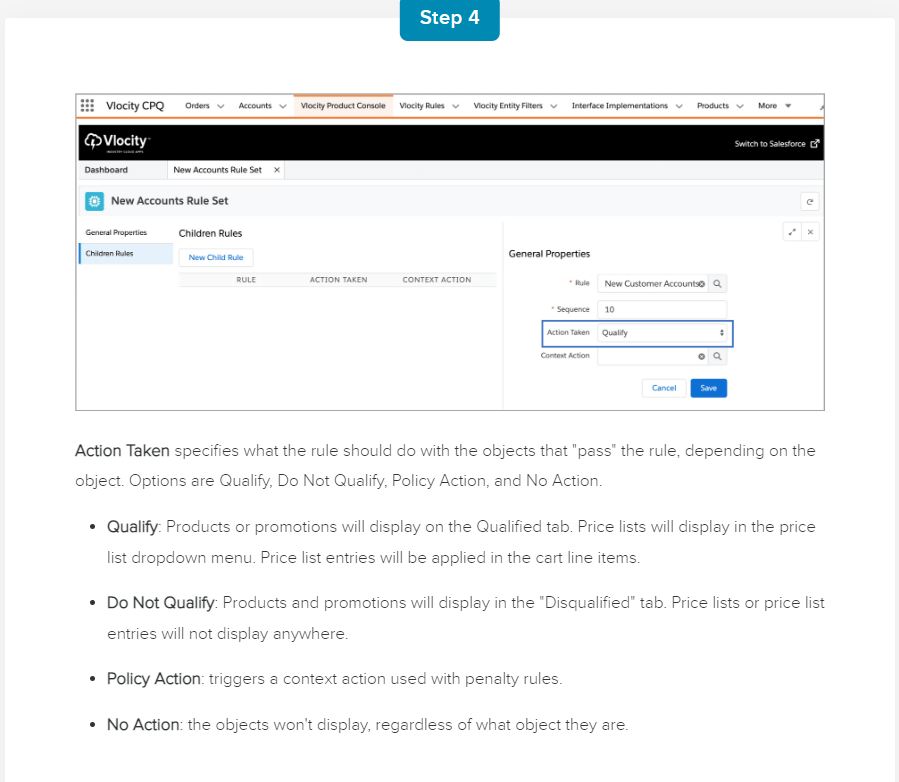
1. **Rule Type:**  
   Options are Qualification, Penalty.
2. **Rule Purpose:**This field currently is not functional but is planned for future use.
3. **Expression Mode:**Options are And, Or, Custom, If Else If, If.
4. **Failure Message:**  
   This message will display in the Disqualified subtab of the Cart.
5. **Header Rule:**  
   Use to optimize multi-rule performance in the evaluation process. This rule is evaluated first and if it fails, the entire rule set fails.

**Adding Children Rules**







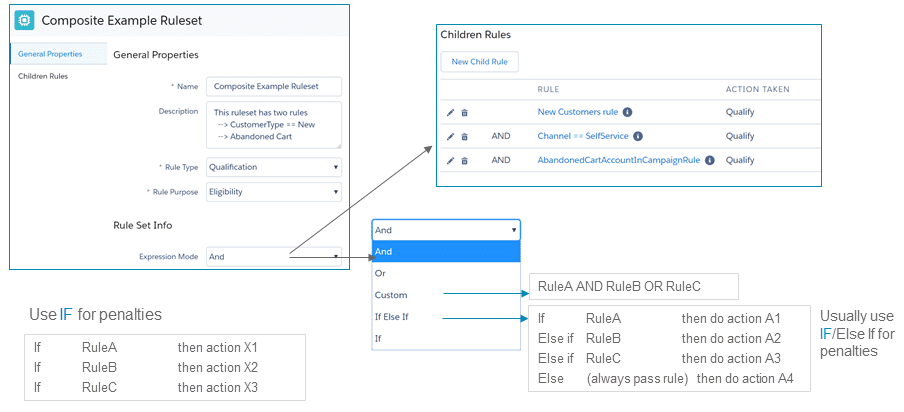


**Expression Modes**

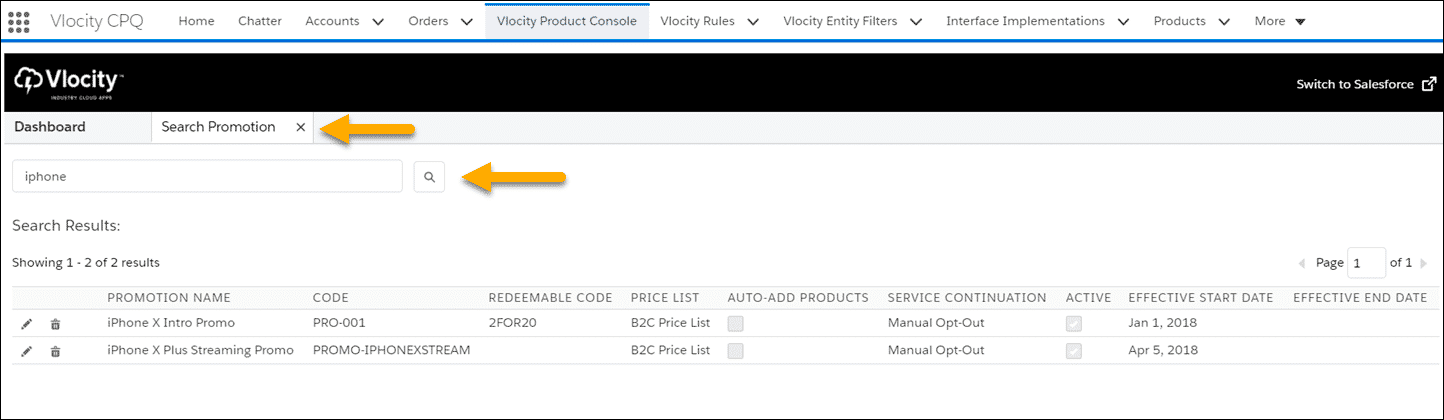
Rule evaluation uses an expression mode (e.g. And, Or, Custom) to compile conditions into a logical expression. Rule evaluation determines the qualification or disqualification of products or promotions.

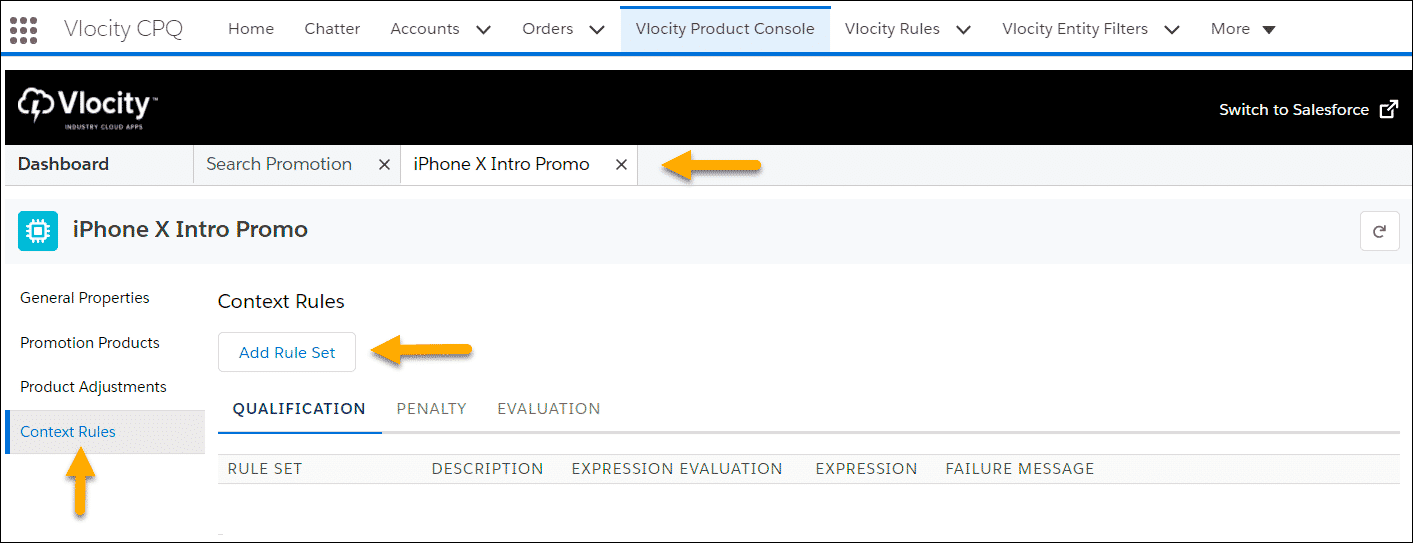
A rule set expression mode is based on one of the following conditions.

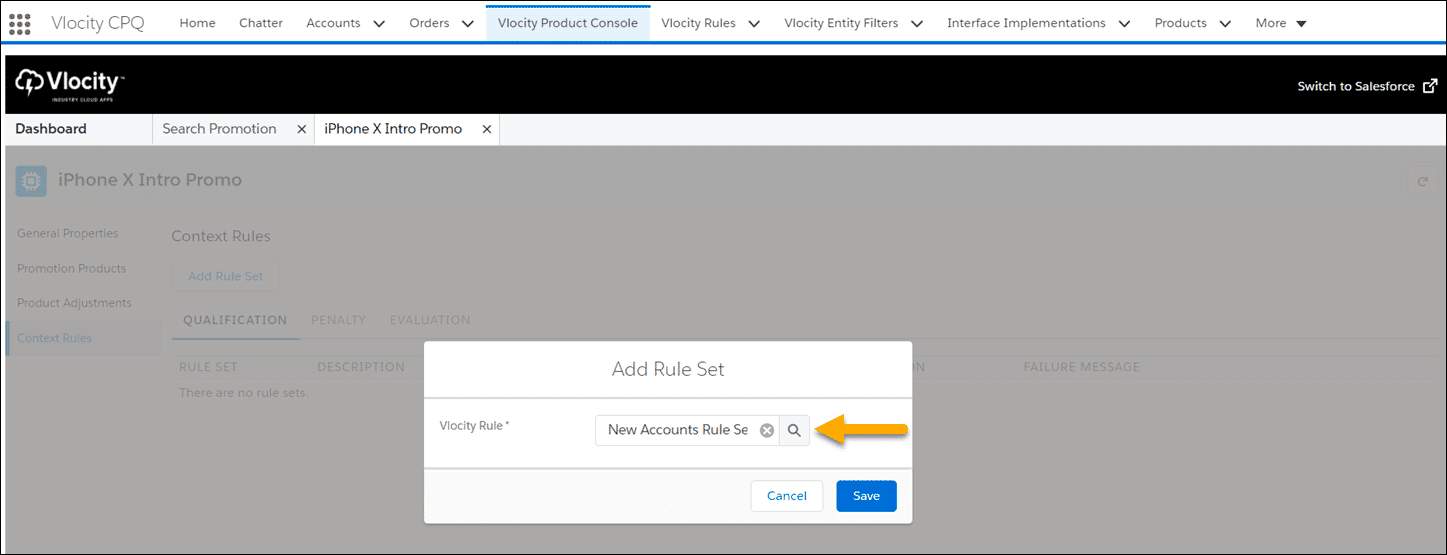
* **And:**Requires that all of the rules in the rule set pass for the rule set to evaluate to Pass. This is the default and is used only with qualification rules.
* **Or:** Requires that at least one of the rules in the rule set passes for the rule set to evaluate to Pass.
* **Custom:** Requires a custom expression to evaluate the rule set. For example, to set the conditions as ((RuleA AND RuleB) OR (RuleC AND RuleD)).
* **If Else If:** Tests the first rule in the rule set, and tests the subsequent rule only if the first rule does not pass.
* **If:** Tests the first rule in the rule set.

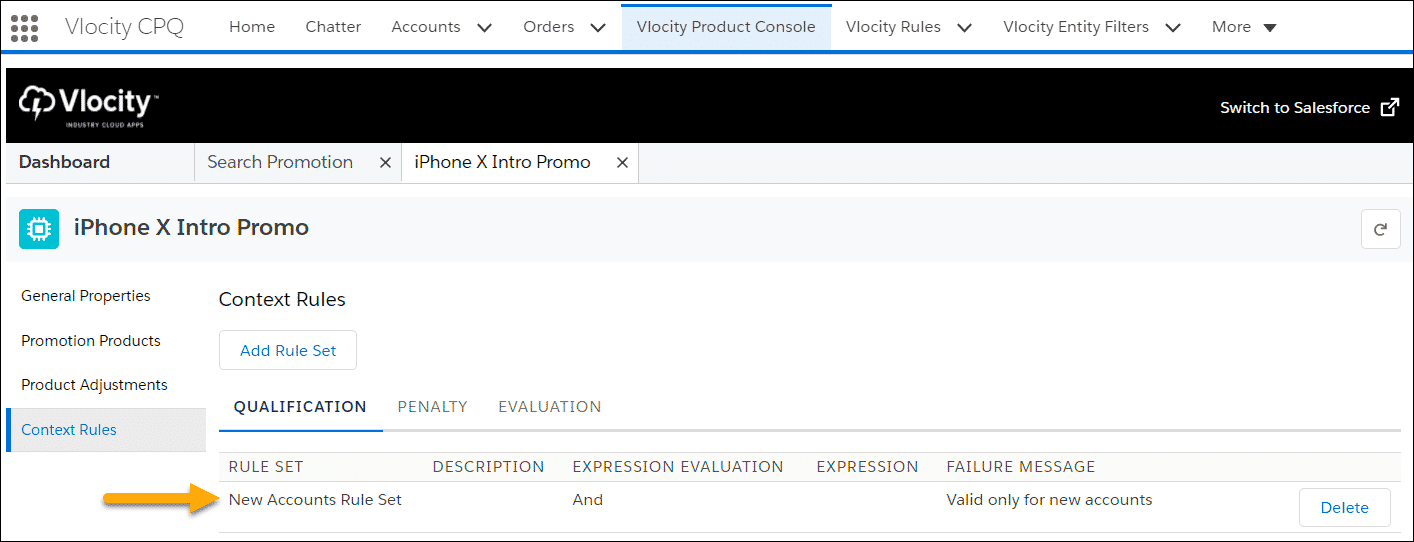


**Applying Rule Sets to Products or Promotions**



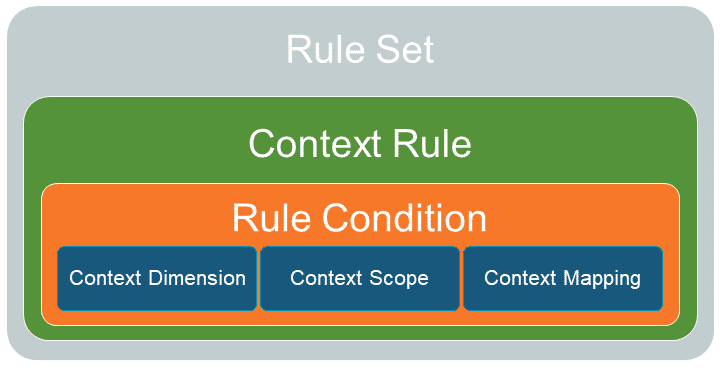






# Context Rules and Conditions

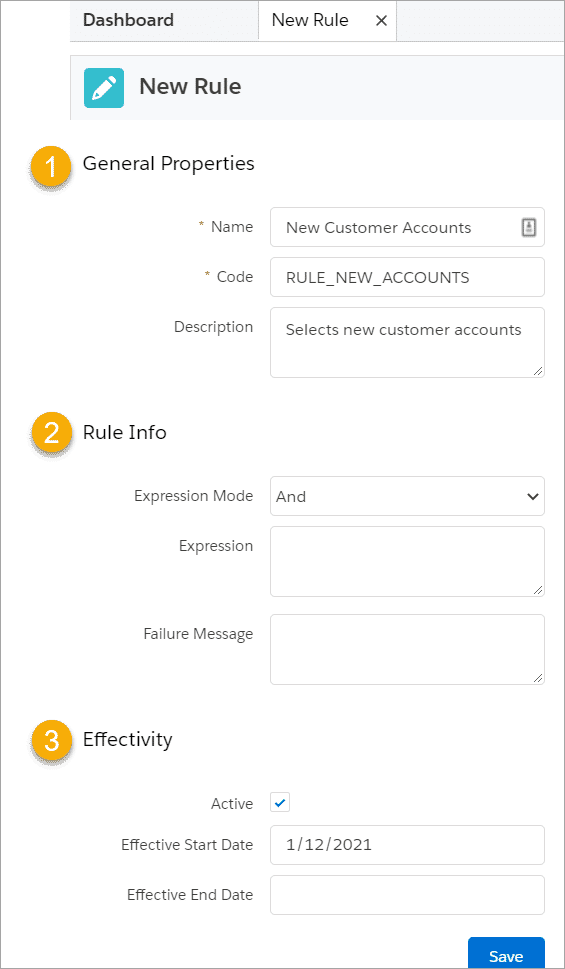
**What Does a Context Rule Contain?**



A context rule contains the information needed to determine when a rule set should run in the Cart.

* Rule condition
* Context dimension
* Context scope
* Context mapping

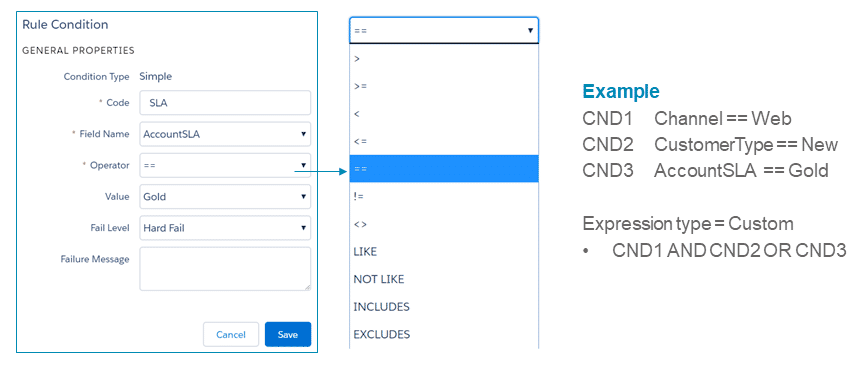
**Creating a Context Rule**



From the Vlocity Product Console Dashboard:  
Under Rules, click **+** next to **Rule**.

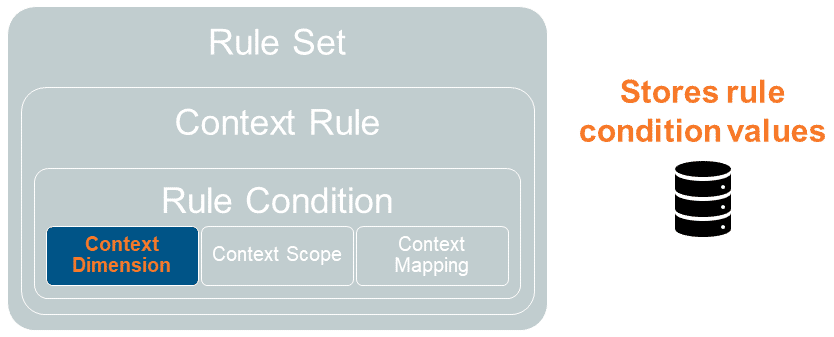
1. **General Properties:**  
   The Rule Code can be any unique value. However, it cannot contain any spaces due to requirements from the Rules engine.
2. **Rule Info:**  
   Expression Mode options are And, Or, Custom. If you choose the custom expression mode, you are required to enter the details of your custom expression in the **Expression**field. There is no need to add a failure message at the Context Rule level, as we will create one on the associated Context Rule Set.
3. **Effectivity:**   
   Although they're not currently enforced, we recommend that you select the Active flag, and that you select the current date as the Effective From date.   
     
   The Effectivity section is common to many records you create from the Product Console Dashboard.

**Rule Condition Operators**



# Context Dimensions

**What's a Context Dimension?**

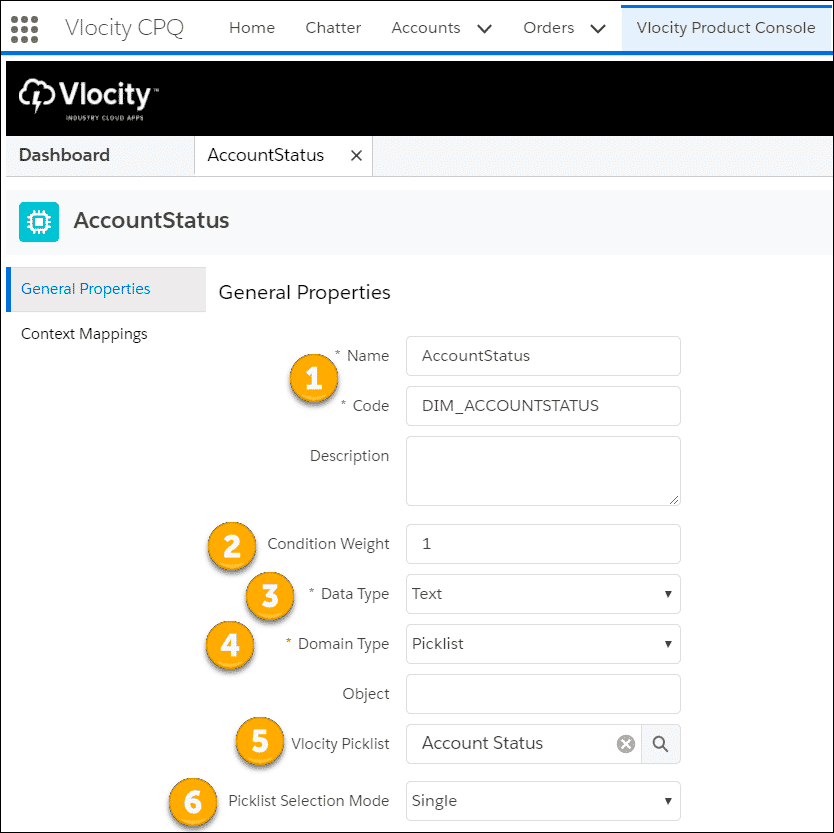


A context dimension is a variable that describes the possible values to use in a rule condition.

You can reuse context dimensions across multiple rule conditions.

The context rules service engine compares the context dimension against data; for example, from an sObject, a function, or a static value defined in the context mapping.

**Creating Context Dimensions**

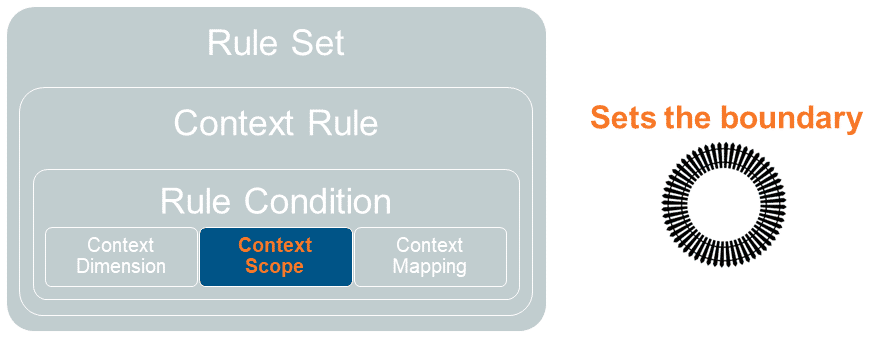


From the Vlocity Product Console Dashboard:  
Under Rules, click **+** next to **Context Dimension**.

1. **Name and Code:** Context dimension names and codes cannot contain any spaces. The rule code is used as an unique identifier.
2. **Condition Weight** is used for price list entries only. The context rule service engine uses this weight to determine the best price to apply when there are multiple prices for which a customer is eligible.
3. **Data Type** options include Text, Number, Date, DateTime, and Boolean.
4. **Domain Type** options include Picklist (Vlocity Picklist), Object Lookup (sObjects), and Type in.
5. **Vlocity Picklist:**Use the lookup to select a picklist.
6. **Picklist Selection Mode**options are Single, Multiple.
   * Single allows users to pick a single value from your picklist.
   * Multiple is reserved for future use.

# Context Scopes

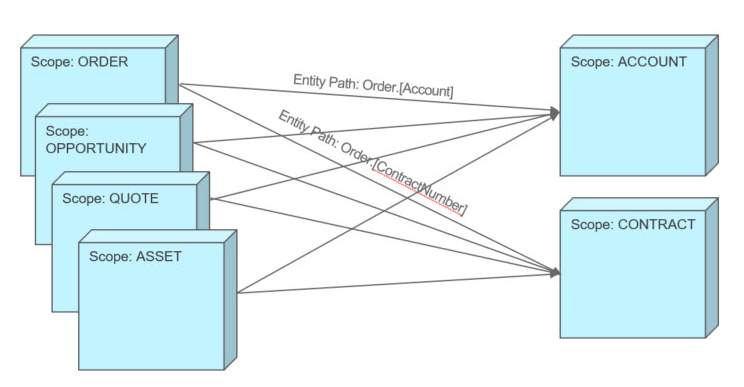
**What's a Context Scope?**



A context scope describes the relational path from a root sObject, such as an Order, to related sObjects, setting the boundaries for the rule function.

Context scopes represent an sObject from which the system retrieves data used by the evaluation logic of rules to determine the output.

**Supported Context Scopes**



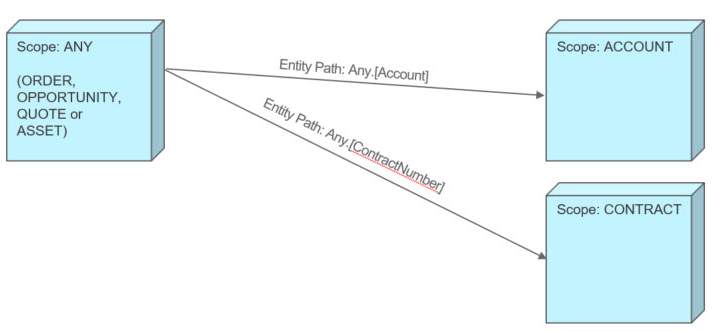
Salesforce Industries supports the following root context scopes:

* Order
* Opportunity
* Quote
* Asset
* Any

Salesforce Industries also supports Account and Contract context scopes, which use the root context scopes to determine the entity path to the required data element.

**Note:**The Asset context scope can only be used for rules invoked during repricing.

**The Any Context Scope**



The Any context scope:

* Is a wildcard context scope that represents all root context scopes.
* Allows you to create Account and Contract scopes that are agnostic with regard to the root context scope. For example, instead of creating separate Order.Account and Quote.Account scopes, you can create a single Any.Account scope that locates the associated account for all root entities, including orders and quotes.
* Provides a flexible and efficient way to define scopes.

**Best Practices for Context Scopes**

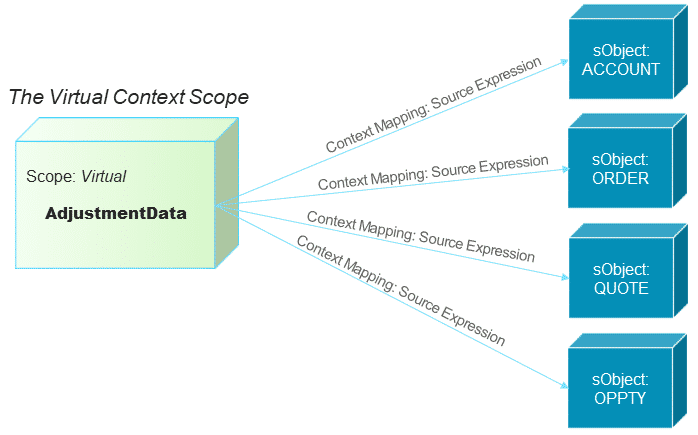
* Use the root context scopes to limit and optimize performance.
* While the Any context scope allows more flexibility and efficiency when defining scopes, in certain cases, when you know that the value can come from only one sObject, consider specifying the object as the root scope entity. By doing so, you limit the number of fields that are retrieved and stored in the org cache. In other words, when the context scope is Order.Account, the associated rules engine retrieves fields from the account sObject only.
* If the context scope is Any.Account, the associated rules engine retrieves fields from all root entities: Order, Opportunity, Quote, and Asset. For example, the Originating Channel field only exists on the Order object. If you specify Any as the root entity, the system will still look into all root entities.

**Setting Up Context Scopes - A One-Time Configuration**

Create context scopes before you define context rules in your environment. This is a one-time configuration that is required as part of the initial context rule setup.

From the Vlocity Product Console Dashboard: Under Rules, click **+** next to **Context Scope**.

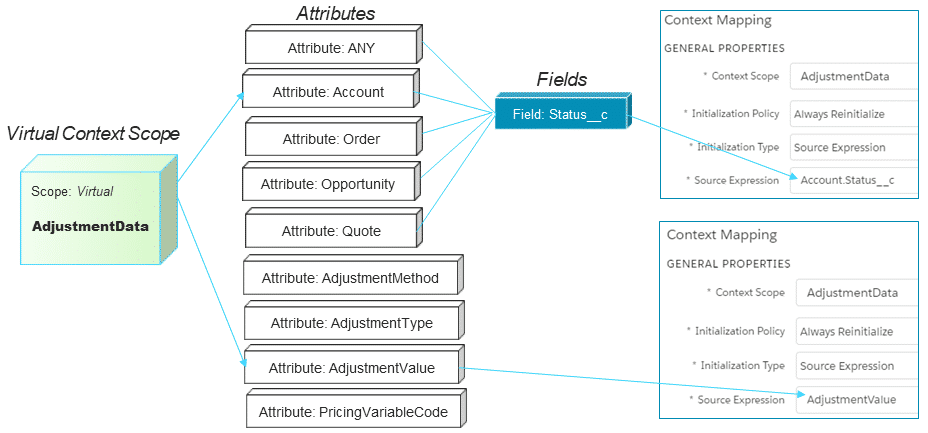
**Virtual Context Scopes**



Virtual context scopes enable you to create context mappings to virtual objects. Virtual context scopes and virtual objects enable you to evaluate data entered by the user at run time that is stored only in memory, such as pricing adjustment data.

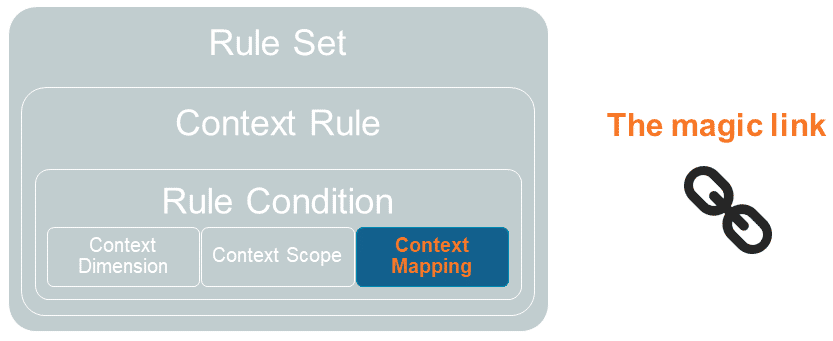
When you create context rules for pricing adjustments, you can create context mappings to virtual object data using the source expression property.

* Map to attributes on the virtual object.
* ANY, Account, Order, Quote and Opportunity and their fields are the only sObjects currently supported. (But you can’t access data from related sObject like standard context mappings.)
* Use the attribute names when writing source expressions.
* For the ANY, Account, Order, Quote and Opportunity attributes, you can refer to fields on those sObjects using the format of [attribute name].[API field name].



# Context Mappings

**What's a Context Mapping? (The Magic Link)**

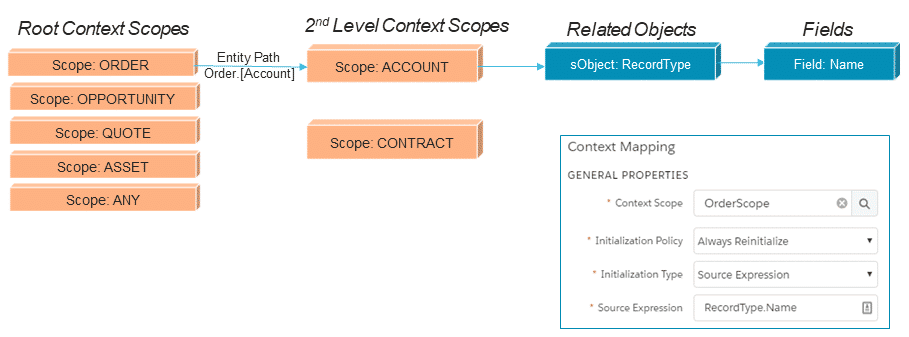


A context mapping uses the context scope to identify the fields on sObjects or computed data to match against context dimension variables for evaluation by the Context rules service engine.

It allows you to map context scopes, such as sObjects, with context dimensions, such as fields or calculated formulas. This way, you create a path to the data for the context rules service engine, which forms a magic link between the data and the rules engine.

Context mappings are held in the org cache.

**Supported Source Expressions**



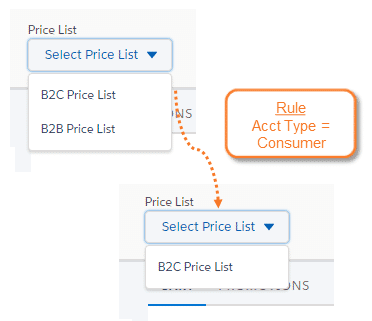
You can trace the object hierarchy using source expressions to access data in objects related via Salesforce lookups.

* Use API names when writing source expressions
* When referring to a field on a custom object using a custom lookup, use \_\_r instead of \_\_c as the suffix syntax, e.g., ContractId\_\_r.ContractTerm using the Asset context scope.

# Using Context Rules with Price Lists

In the same way that you can use a qualification context rule to display qualified products and promotions in the Cart, you can use qualification context rules to control the display of price lists in the Cart's price list menu. You can also assign context rules to child price lists.

* When a context rule is used with a parent price list, it controls what price lists the user will see in the Cart price list dropdown menu. All price lists that qualify will display as available price lists, with no condition weighting (which we will cover in the next lesson).
* When a context rule is used with a child price list, it is used to determine which price to apply to products in the cart based upon the rule condition criteria. The price list entry for the child price list will be automatically applied to the line item.

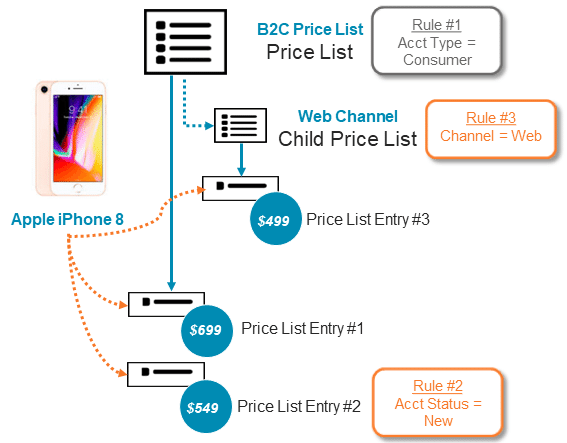


**Applying Specific Prices to Products in the Cart**

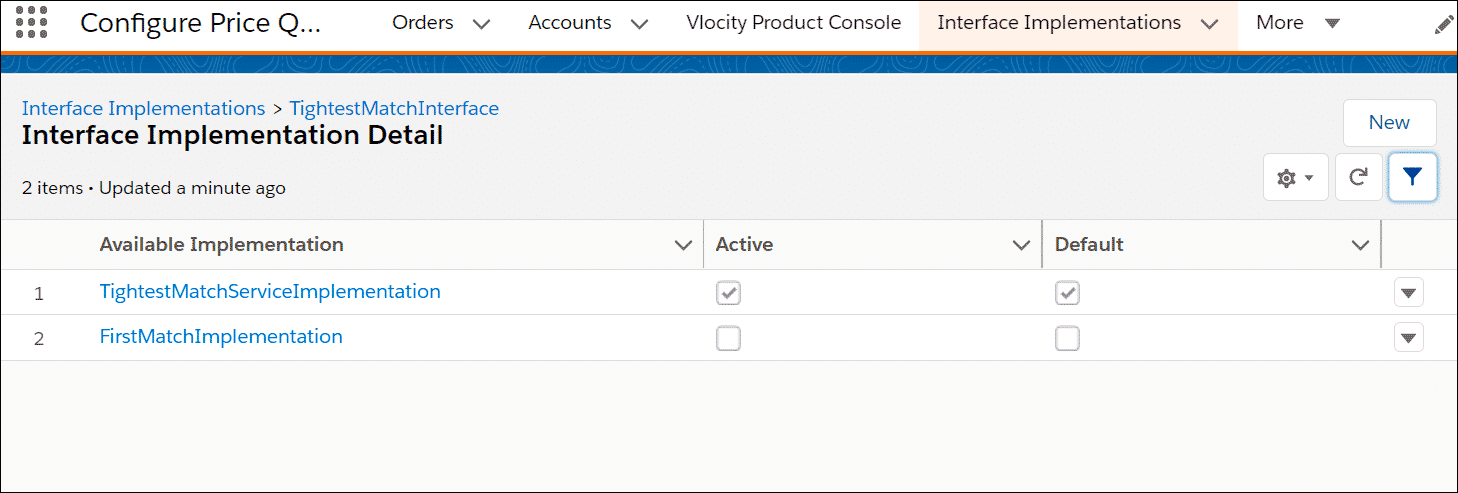
You can apply specific prices to products in the Cart based upon factors such as customer segment, account status and location.

As we explain in the Pricing course, products can have multiple price list entries across price lists or within a single price list. You can use context rules to determine which price list entry to apply when the product is added to the Cart. For example, you might want rules like this:

1. Consumers only see the B2C Price List.
2. New customers receive certain prices.
3. Orders placed over the web receive certain prices.

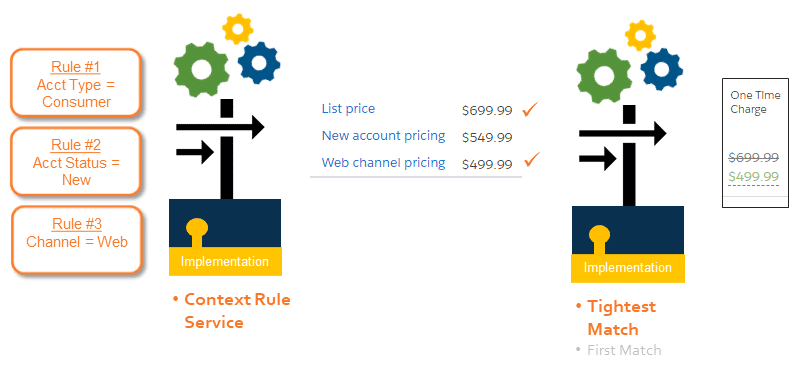


# Tightest Match



**What if the customer is eligible for multiple prices?**  
In the end, a single price must be set. That’s when the **TightestMatchInterface** comes into play.  
This interface includes two implementations in the managed package:

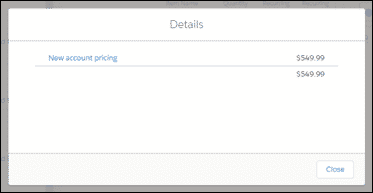
* **TightestMatchServiceImplementation** uses context dimension condition weights in order to determine the “tightest match” price.
* **FirstMatchImplementation**ignores those weights and selects the first match that it finds.



**Tightest Match – 1 Match**

* Displays as standard price list entry
* Details window displays “winning” price list entry display text and price



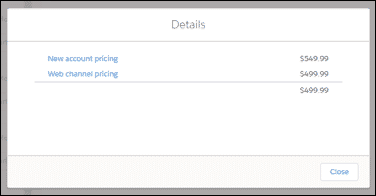


When there is only one “winning” tightest match, the user experience is the same as a standard price list entry.

**Tightest Match – 2+ Matches**

* Most recent price list entry “wins”
* Strikethrough “losing” price list entry
* Details window displays all “tightest matches” price list display text and price





When there are multiple “winning” tightest matches, the most recent price list entry is selected, and the price list entries not selected will display in the price Details window.

**Calculating Condition Weights**

Condition Weights are calculated from context dimensions for each rule condition assigned to the child price list (if it exists) + the price list entry. A total weight is assigned to each price list entry.



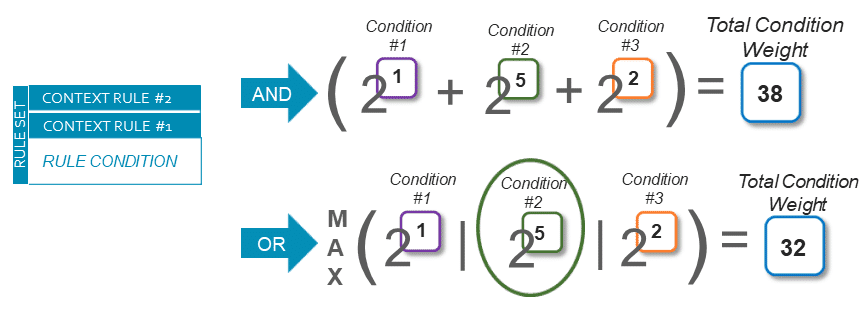
When you add a product to the Cart, the rules engine determines all of the qualified child price lists and price list entries based on the configured rules. Then, it assigns a weight to each qualified price list entry, roughly using the formula shown in the diagram above. The algorithm uses bitwise operators for maximum processing speed, so the weights are calculated as 2n.

**Calculating Multiple Condition Weights**

A context rule can have multiple conditions.  
Depending on the expression mode, the conditions are joined together using AND or OR.

Tightest Match calculates multiple conditions as follows:

* AND rule condition weights are summed together
* OR rule conditions select the highest weighted condition value



**More on Condition Weights**

Additional information to consider when you create condition weights:

* Condition weights are only used when evaluating price list entries. Since context dimensions are reusable, a context dimension might be used for rules for both price list entries and for products or promotions. The condition weight value will be ignored on a context dimension used in a rule for a product or a promotion.
* Condition weights can be any whole number between 0 and 60.  
    
  Since null, 0, and >60 condition values and price list entries with no context rules default to 1, leaving the condition weight blank for non-weighted context dimensions is most often the right approach.  
    
  Then, when you want to weight conditions, since condition weights are calculated as 2n, it is easy to weight conditions quickly since 21=2, 22=4, 23=8, 24=16, and so on.

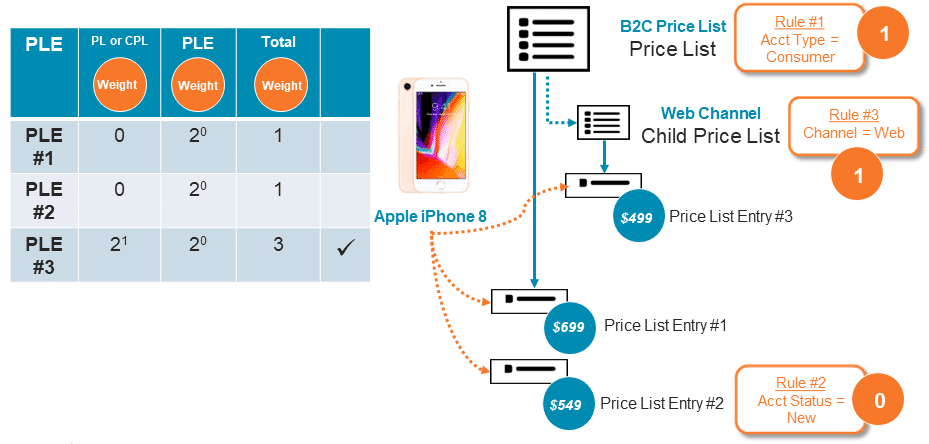
**Algorithm Assumptions**

* Price List Entries:
  + With no context rules default to 1
  + With a 0/null weight default to 1
  + With values > 60 default to 1
* Weights for price list entries on a Child Price List will be higher than a price list entry on a Parent Price List
* Parent Price Lists with a rule default to 0
* Price list entries on a child price list combine the price list entry weight + price list weight

**Total Weights**

After all conditions have been assigned a weight, the rules engine sorts all the price list entries by weight, and then returns the price list entries with the highest weight.

If there are multiple price list entries with the highest weight for the same pricing variable (for example, One Time Charge or Monthly Recurring Charge), the interface selects the most recently created price list entry, and applies that price to the variable in the line item.



In the above scenario, let’s assume that all rules pass.

* Account Type = Consumer
* Account Status = New
* Channel = Web

Weights are calculated as follows (PLE = Price List Entry):

**PLE # 1**

* Parent B2C Price List = 0 (A rule assigned to a parent price list default to a 0 value)
* PLE #1 = 1 (PLEs with no rule default to a 0 value, or 20)
* TOTAL = 1

**PLE # 2**

* Parent B2C Price List = 0 (A rule assigned to a parent price list defaults to a 0 value)
* PLE #2 = 1 (PLEs with a zero weight rule or 20)
* TOTAL = 1

**PLE # 3**

* Web Channel Child Price List = 21 (which is 2)
* PLE #3 = (PLEs with a zero weight rule or 20)
* TOTAL = 3