Salesforce CPQ and its product rules are designed to help sales reps get the right products on the quote the first time around. Think of product rules as guardrails, there to help sales reps stay on course when navigating complex combinations of products and services. As an admin, putting a small amount of effort into creating product rules can save huge amounts of time and money for your business, while keeping your customers happy.

**Product Rules for All Occasions**

Product rules come in four flavors. Let’s look at the different types and the different ways you can use them to help your sales reps.

| **Type** | **Purpose** | **Example** |
| --- | --- | --- |
| Alert | Shows a message to your sales reps about a potential issue but allows them to ignore it. | Reminds sales reps of an upsell opportunity when they add a specific product to the quote. |
| Validation | Shows a message to your sales reps about a problem they must fix before they’re allowed to save the configuration or quote. | Tells sales reps they’ve selected the wrong toner for the printer they’re trying to add and stops them from saving the illegal configuration. |
| Selection | Automatically adds, removes, or hides products during bundle configuration. Also, automatically adds products to a quote. | Pre-checks options in a bundle based on the account’s industry type; unchecks and hides options known to be incompatible. |
| Filter | Shows only specific products in a feature that uses the Dynamic selection option. | Lists only products with a product code that contains the letters “cable” in a mini product selection page. |

When CPQ admins talk about product rules, they usually abbreviate the name of the rule. For example, they frequently shorten “validation product rule” to just “validation rule.” It’s OK to call it that. Just remember that it’s really a product rule of a certain type, and you go to the same tab in Salesforce to create all types of product rules.

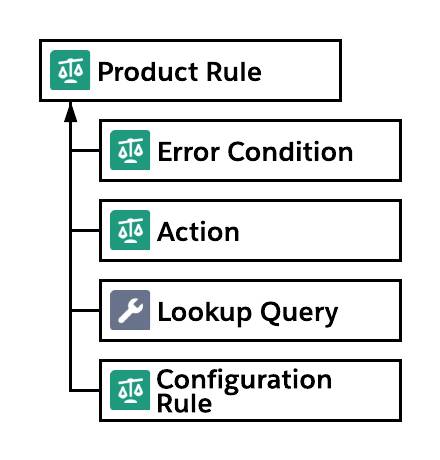
## Product Rules Are a Team Effort

CPQ administration usually requires you to create or update records of some sort, often across multiple objects. Product rules are no different. Each object related to product rules has a specific role to play in the show we call “Quote the Right Thing.” Let’s meet the cast of characters, starting with our star performer.

**Product Rule**

The phrase product rule has two meanings. First, it’s meant to describe the concept of business logic that helps sales reps get the right products onto the quote. Second, it refers to the actual object named product rule.

A product rule record has a few details about how the rule should behave, but mainly it acts as a container, or point of contact for any records on other objects related to the rule. There are four other objects that look up to it. A product rule can technically work without related records, but it won’t do much. To really make the best of product rules, you must include at least one of the other (following) related objects.



**Error Condition**

Simply put, error conditions control when rules trigger. With the right error conditions, your rules will only run when they need to.

**Action**

Actions are records that hold the instructions for how CPQ should make changes to the way a bundle is configured. They’re mostly used for selection rules, but have a small role in filter rules too.

**Lookup Query**

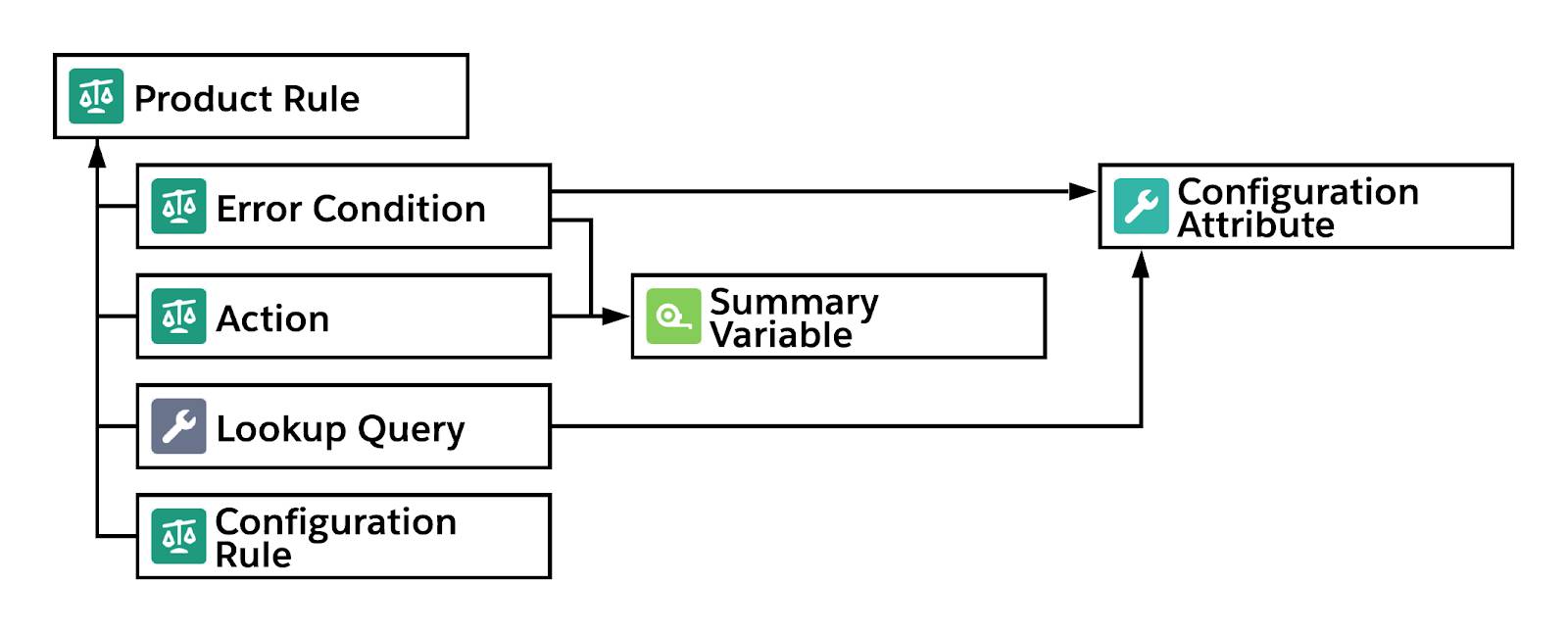
Sometimes businesses have many product rules that differ by only a little bit. In that scenario you can create one rule that looks to a data set to drive behavior. Lookup queries tie a product rule to the object that houses the data set.

**Configuration Rule**

You create a configuration rule to tell CPQ that a product rule applies to a specific configurable bundle. If you need the rule to apply to more than one bundle, you just create more configuration rule records.

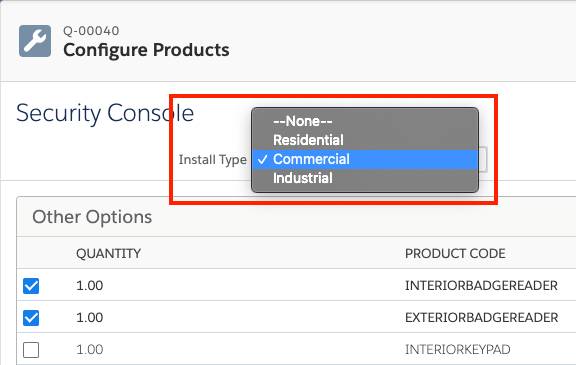
## Other Contributors

In addition to the four main objects related to product rules, there are a few other objects that often contribute to how product rules behave.



**Configuration Attributes**

Configuration attributes take the form of a field that sales reps can set during product configuration.



Although configuration attributes are not directly related to product rules, they’re often used as part of error conditions, and sometimes as part of lookup queries.

**Summary Variable**

A summary variable is another CPQ-specific tool sometimes seen in product rules, and it does one simple task: add up values across many CPQ-related records. It’s a little like a rollup summary field, but with some advantages specific to CPQ. Summary variables are often used in error conditions, and sometimes actions.

So those are the objects that can possibly contribute to a product rule. Some rules use only a few objects, and others use all of them. By the time you’re done with this module, you’ll have created rules that use all parts.

## Under the Right Conditions

An error condition is basically a test, the pass/fail kind of test. It looks at some data (like a quote line’s product code) and compares it to other data (like the string “CLOUDSTORAGE”). In this example, it checks if they’re equal. If so, the test passes and the rule is one step closer to running.

## Summary Variable

The job of a summary variable is simple: Look at a set of data, and summarize it down to a single numeric value. For example, you can:

* Count the number of all “WiFi Access Point” assets on an account.
* Find the maximum discount percent across all lines on a quote.
* Sum the quantity of selected options that have a product code that ends in “MIC” within a bundle.

**Limits of Testing Product Option Fields**

When you create an error condition for a bundle-specific rule you can test summary variables, configuration attributes, and product option fields. For example, you could check the product option field of Quantity to see if it’s greater than 10. The way CPQ handles tests involving product option fields is to cycle through each selected option, performing every test against the option as if it were the only one that exists. This method holds within it two important caveats to remember when making bundle-specific error conditions.

First, CPQ only tests *selected* options, ignoring anything that’s unchecked. That means it’s impossible to test for something that *isn’t* selected. That’s why you created a summary variable in the beginning of this unit, so that you could discover if zero microphones were selected. If you didn’t have the summary variable, and you just tried to test if the Product Code field does not contain “MIC” then you would’ve got a bunch of false positives. For example, the keyboard option doesn’t contain “MIC” so the condition would be true, and the rule would fire, even if an actual microphone option was selected.

The second caveat involves testing two separate options in the same rule. For example, imagine you want to test a coffee bundle to see if both “Cream” and “Sugar” options are selected, using these conditions.

* Product Code = Cream
* Product Code = Sugar

Because you want both to be true, you need to use “All” for the Conditions Met field. When CPQ gets to the Cream option and performs the tests, the Sugar test will fail. It’s impossible for a single option to be both Cream and Sugar, so the rule will never run.

The workaround is to use a summary variable to count one of the options, such as Sugar. That would give you these two tests.

* Product Code = Cream
* Sum of Sugar Options > 0

Now when CPQ gets to the Cream option, both tests can pass, and the rule can fire.

Creating good logic for product rules is sometimes a challenge, so the most important thing you can do after creating a rule is to thoroughly test it. Try testing when you expect the rule to fire, as well as scenarios when it shouldn’t.

Now that you’ve seen how rules work in the context of bundles, in the next unit you investigate selection rules, which are used heavily (but not exclusively) in bundles as well.

**Product Configuration vs Quote Line Editor**

Sales reps who use CPQ to create quotes spend a lot of time on three pages.

* Product Selection: where they choose which products to add to the quote
* Product Configuration: where they configure bundles
* Quote Line Editor: where they see pricing and apply discounts (among other things)

A product rule can run on either the Product Configuration page or on the Quote Line Editor, but not both! It’s rare that the same business logic needs to be in both places, but if you find yourself in that situation, you have to create two separate rules.

## Evaluation Event

If the Scope field tells CPQ where the rule should run, then the Evaluation Event field helps tell it when it should run. There are five times that CPQ can evaluate rules as a sales rep moves through the quoting process. The combination of your Scope and Evaluation Event field choices determine the timing.

| **Scope** | **Evaluation Event** | **Occurs When the User...** | **Applicable Rule Types** |
| --- | --- | --- | --- |
| Product | Load | First loads the Product Configuration page. | Selection, Filter |
| Product | Edit | Changes specific configuration attributes or options on the Product Configuration page. | Selection, Validation |
| Product | Save | Clicks the Save button on the Product Configuration page. | Alert, Validation |
| Quote | Save | Returns to the Quote Line Editor after selecting or configuring a product. | Selection |
| Quote | Save | Clicks the Save or Quick Save buttons on the Quote Line Editor. | Alert, Validation |

The tricky part is that CPQ avoids running some rule types at certain times. For example, CPQ does not evaluate Alert rules when the user first loads the Product Configuration page. So, choosing a Scope/Evaluation Event combination of Product/Load means CPQ will never evaluate your Alert rule, and therefore never run it! The last column in the table above shows which rule types are evaluated for every point in time.

Finally, you can set the Evaluation Event to “Always,” but be careful, this value is a little misleading. “Always” doesn’t mean a rule is always evaluated. Instead, it’s better to think of the “Always” value as meaning “All Applicable.” Take a look at the first row in the chart again. You can see that CPQ will NOT evaluate Validation rules when the Product Configuration page first loads. So, if you choose “Always” for the evaluation event of a Product-scoped Validation rule, it will only be evaluated when a user makes edits or clicks Save on the Product Configuration page.

The “Always” event is a safe choice if you’re not sure which other event to choose. But if you find that your rule is a bit too overactive, consider choosing a specific event.

## Don’t Wait, Apply Immediately!

Product rules that use “Edit” for evaluation event are meant to provide real-time feedback to sales reps. Reps make a change, a rule runs, and they see something happen. This is a great feature, when used in moderation. Too many rules running too often makes for a bad user experience.

Thankfully, rules with the Edit event don’t actually evaluate every time an edit occurs, only when certain edits occur. And you get to decide which edits call for evaluation. For example, you might have a bundle with 20 options, but only one is tied to a rule. In that case, you'd only want the rule to run if that one specific option is selected or unselected (or in other words, edited). To tell CPQ to evaluate "Edit" rules when a specific option is selected or unselected, check the "Apply Immediately" checkbox on the option record.

Attributes are often used in product rules as part of the condition logic. If you want CPQ to evaluate a rule the moment a sales rep changes the attribute value, make sure the rule's evaluation event is "Edit" then check "Apply Immediately" on the configuration attribute record.

Finally, there’s one last way to tell CPQ to evaluate all edit-event product rules: the Apply Rules button. This allows your sales reps to make many changes that CPQ might need to evaluate, then CPQ evaluates all edit-event rules at once.

Some customers use the Apply Rules button as an alternative to apply immediately as a way to reduce how often validation rules trigger. Relying only on Apply Rules could be dangerous if you have Validation rules set to the Edit event, since there’s a chance they never run. If you use Validation rules without Apply Immediately, at least set the evaluation event to Always. That way, clicking Save will evaluate your important Validation rules as a final safeguard.

Also be careful when using Apply Rules as an alternative to apply immediately for selection rules, since selection rules are never evaluated when Save is clicked. You can easily end up with selection rules that don’t run. Only consider this setup if your selection rules are used for guidance, and not if they’re used to ensure technically valid configurations.

## Evaluation Order

There’s one last thing that can affect when CPQ evaluates your rules: Evaluation Order. This optional field allows admins to sequence selection type rules that check or uncheck options in a bundle. For example, you can have a general-purpose rule that preselects common options for all customer industry types. Then, you can have a second specialized rule that unselects a few options for just one industry.

In that scenario, you want the general rule to run first, otherwise it rechecks options you actually want unchecked.

To avoid that issue, just set the Evaluation Order of the general rule to 10, and the specialized rule to 20.

Rules are evaluated in numeric order, so 10 first, then 20, and so on. We recommend that you use increments of 10 when ordering your rules. That way, if you need to insert a rule between 10 and 20, you can just number it 15 and not have to renumber everything after 1.

Evaluation Order, along with Scope and Evaluation Event, give admins a huge amount of control over when and where CPQ evaluates rules. In the next unit we look more closely at rules related to bundles.

## The Helping Hand of Selection Rules

So far we’ve spent a lot of time talking about product rules that prevent sales reps from making mistakes by alerting them of potential problems. But what if there was a kind of rule that took action to automatically make necessary changes to a configuration to avoid issues in the first place? Well that’s exactly what selection-type product rules aim to do.

For simplicity, we just call them selection rules from this point forward. There are a number of ways selection rules take action to affect how bundles are configured.

| **Action** | **Result** |
| --- | --- |
| Add | Checks an option |
| Remove | Unchecks an option |
| Disable | Makes an option’s checkbox grayed out and uncheckable |
| Enable | Makes an option’s checkbox normal and checkable |
| Disable & Remove | Unchecks an option, and makes the option’s checkbox grayed out and uncheckable |
| Enable & Add | Makes an option’s checkbox normal and checkable, and checks it |
| Hide | Completely removes an option from the page layout |
| Show | Brings a hidden option back into the page layout |
| Hide & Remove | Completely removes an option from the page layout and unchecks it |
| Show & Add | Brings a hidden option back into the page layout, and checks it |

With the right combination of actions, a single product rule can quickly get a sales rep to a properly configured bundle, or at least give them a head start down the right path.

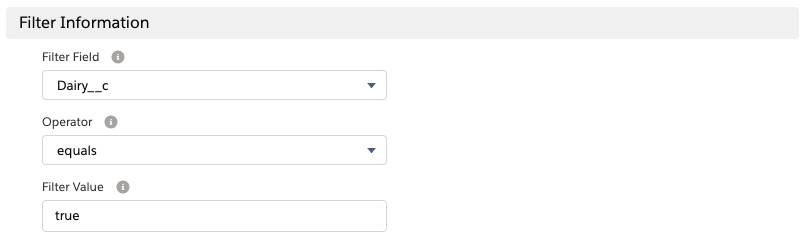
## One Action to Rule Them All

Selection rules use action records to tell CPQ how to change options during a configuration. In the previous unit, each of your rules acted on just one product. But what if you needed to perform the same action on multiple products? For example, imagine you’re a barista and you know that your customer is lactose intolerant (thanks to information on their profile). You would want to disable all dairy options in your coffee bundle so that you can’t accidentally include skim milk, 2% milk, whole milk, or cream.

As you create the selection rule you can create a separate action to disable each product. Unfortunately, later when you get a new dairy product that needs the same treatment, you have to go back to the rule and create another action. Overall, one product per action is not fun to maintain.

Thankfully there’s another option as long as you have a way of identifying which products contain dairy. For example, you can make a checkbox named Dairy\_\_c on the product record, then check each dairy product.

Then, you would create a single action for your rule. Instead of filling in the product lookup, you’d use the fields in the Filter Information section to look for any products where Dairy\_\_c equals true.



That’s all there is to it. No dairy products allowed, no upset tummies for your lactose intolerant customers. Later when you add another dairy option to the bundle, you just need to make sure your Dairy\_\_c flag is properly set on the product, and the existing rule automatically disables it.

Finally, notice that you didn’t use the Product lookup field on the action record. If you use both lookup and filter information, only the lookup product is affected. Just leave it blank if you use a filter.

## Add Products to a Quote with Selection Rules

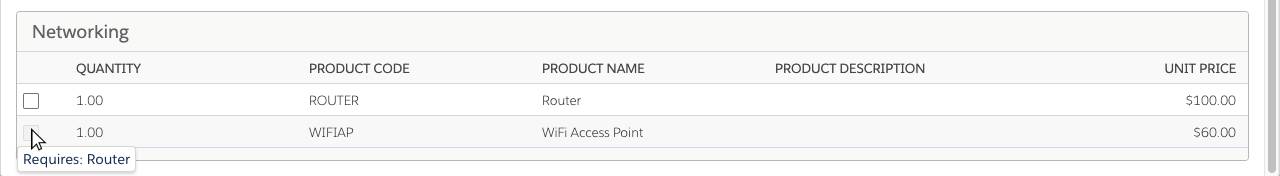
Selection rules are often used to control how bundles are configured, but they have another purpose outside the context of bundles. You can also use a selection rule to add a product directly to the quote. For example, you might want to give customers a free, branded USB thumb drive as a promotional item whenever they buy an expensive monitor. In this example, none of the products are bundles.

There are a few differences between product- and quote-scoped rules,

Quote-scoped selection rules run every time you return to the Quote Line Editor from the Product Selection page or the Configure Product page. So that means if you delete the USB thumb drive from the quote, then add another product, if the 4K monitor is still on the quote, the thumb drive comes back! Conversely, quote-scoped selection rules only run when returning from product selection or configuration. So you could delete the thumb drive, and it will stay gone as long as you don’t add more products or reconfigure an existing product.

**Selection Rules Versus Option Constraints**

An option constraint is a simple tool that behaves a little like a product rule, used only in the context of bundles. It ties two options together, so that one can only be checked if the other is selected. This is a little like a selection rule that has an enable action. In the screenshot, you see WiFi Access Point is dependent on the Router using an option constraint.



Option constraints can also tie two options together, making it only possible to choose one or the other. So clicking one disables the other. This is a little like a selection rule that has a disable action.

Option constraints are easier to set up than product rules, but there are a few good reasons to use product rules anyhow.

* A single product rule can be applied to unlimited bundles, whereas an option constraint is specific to a single bundle.
* Product rules can select options with Add actions. Option constraints can only enable an option at best.
* Product rules can have multiple conditions that evaluate information outside the context of the bundle. An option constraint has one simple condition: if option X is selected.
* Product rules can act on multiple options, but an option constraint is limited to just the one.

**Selection Rules Versus Validation Rules**

There’s often more than one way to keep sales reps on the right path to a properly configured bundle. For example, if your goal is to prevent dairy products from getting to lactose intolerant customers, you have a few choices. One is to create a selection rule to remove the offending products. Or you can create a validation rule that pops up an error message if there’s a dairy product selected when the sales rep clicks Save.

When you design your product rules, consider what kind of experience your want for your sales reps. Then, get their feedback after they pilot the new rules. Being responsive to feedback leads to sales reps adopting CPQ faster, which is a great goal to aim for.

Selection rules are a great way to guide your sales reps to a quote that is complete and correct. In the next unit, you learn how to create a single product rule that uses data in a custom object to drive its behavior.