TEAMCENTER

Product Configurator on Rich Client — Usage

Teamcenter 2412

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow’s idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: **support.sw.siemens.com**

Send Feedback on Documentation: **support.sw.siemens.com/doc_feedback_form**

Contents

1. About Product Configurator

Products are increasingly becoming more complex and customers are demanding greater individual choice. Teamcenter Product Configurator is used to introduce and maintain variability across the product suite at your site. It allows selection of features that are most important to customers. The variant data is independent of any application domain, for example, CAD design or part planning.

> Example:
>
> Your company makes cars for the North American markets and wants to expand to Europe. For England and Ireland, cars require right-hand steering wheel and transmission controls. You begin by listing these features in a configurator dictionary or configurator context. This way you can define the variability of your entire product suite. You can then reuse the common features across different product lines.

Where do I go from here?

 Administrator	See *Product Configurator — Deployment and Administration*.
 Configurator experts, configurator administrators, or configuration analysts	
I want to perform configuration tasks in Active Workspace	See *Product Configurator on Active Workspace — Usage*.
Create dictionaries, configurator contexts, and construct configurator rules	See the **basic concepts of using Product Configurator**.
Create variable products	Managing a discrete product variant for every possible family combination is inefficient and error prone. Instead, you can choose to leverage the commonality across the product and build the variability into the product data. You manage the variable product definition and derive all valid product variants from it. These tasks are described in the section on **creating variable products**.
Working with Teamcenter objects to specify variability	When you specify the variability to offer on products, you work with families, features, variant rules, and so on. To understand this, see the section on **defining families, features, and configurator rules**.
Introduce and define variability	There are several components in Teamcenter involved in managing variability. You use a *dictionary* to collect all the variant data related to a product suite. You can store *groups*, *families*, and *features* in it and allocate these directly from the dictionary. You can also create

	a *configurator context* instead of a dictionary. Go through the sections on how to **introduce and define variability** to understand this.
Assign variability to a product suite	See the sections that describe how to • **Create a product line family, a model family, or a summary model family**. • Create corresponding product models or summary models. • View and modify the available features using the availability matrix.
Configure and analyze	You can perform configuration using the manual or the guided configuration mode. • The *manual configuration mode* is used to perform a study or an overlay analysis in the application that manages product content. It is also used by configuration analysts to study the system response based on variety of selection paths. See *Configure with custom variant configuration (manual validation)*. • The *guided configuration mode* is used to simulate the configuration behavior for customer interactions. This is to ensure that there is a valid configuration path for every planned product variant. See *Configure with guided variant configuration (active validation)*.

2. Overview of Product Configurator

What is Product Configurator?

Product Configurator enables you to formally introduce and manage the variability you wish to offer to your customers across your product suite.

You can maximize flexibility and reuse by:

- Consistently managing all variability in one central repository.
- Managing one or many corporate dictionaries of all features offered across all product lines.
- For each product line or product model, specifying exactly the subset of features that are relevant and allowed.
- Enabling intuitive and efficient grid-based and table-based user interaction.
- Enabling management of configurator data separate from product data.
- Allowing marketing users, engineers, and even dealers and sales engineers to define, react to, and leverage variability through the product definition process.

Role	Process overview
Define/control the product suite	
 Product managers or product marketing owners	Define and control the overall product suite, such as: • Product lines and product models that are offered to the market. • Features that are relevant for product model. • Rules specifying which features should be offered or restricted in combination for an orderable product variant.
Define/use variability	
 Configurator experts, configurator	Create and maintain the configurator rules base. Configuration experts use Product Configurator to define the variability against a nomenclature with which designers are familiar.

Role	Process overview
administrators, or configuration analysts	
 Engineers and CAD Designers	Aware of the variability their design supports. These users typically need to assert constraints only to express when a given set of features requested by product management or product marketing is not feasible for technical, financial, or productivity reasons.

Product Configurator is supported only on Windows 64-bit systems and Linux 64 platform. This restriction is applicable to the BOM applications that use Product Configurator for variant management, such as 4GD, Structure Manager, Product Master Manager, Manufacturing Process Planner, and Requirements Manager.

Managing variability of your entire product suite

An organization can maximize consistency and reuse of the variability that exists across their entire product suite by collecting all features into one or more corporate dictionaries. These dictionaries are then used as the basis for which product management and marketing declares what set of this variability is valid for a given product line or product model.

In terms of variability, feature families can be thought of as the questions that are asked as a part of configuring a given product variant. For example, “What color do you want?”. Features can be thought of as the allowable answers to those questions. For example, “Given the choices of blue, silver, or white colors, I choose silver”.

The configurator rules that express how these features can coexist in a valid configuration are stated in terms of defaults, that is, suggested features that can be overridden by the user, or feature combinations that must or must not be selected in a combination.

Features and configurator rules that are valid for a particular product line are collected into a configurator context. All of the features and rules collected by this configurator context can exist and be independent of any content, such as designs, parts, processes, partitions, requirements, and documents. For example, you may define the variability of a product at the concept planning stage before you start any CAD designs. Availability of variant data at an early stage in the design process allows CAD designers to concentrate on developing solutions without needing to define variants.

Product Configurator allows you to create comprehensive variant data for various Teamcenter applications in a single location. These definitions are used by other applications, for example, 4G Designer, in which you apply the defined variability to the CAD designs. You can also create items to represent variant feature libraries and products in other applications (typically, in My Teamcenter) and send them to the Product Configurator.

Delivering the product variability your customers demand

Why Product Configurator in Teamcenter?	Teamcenter is a source for the engineering definition of a product. Often the upstream market-driven decisions that drive engineering activities are very disconnected from the engineering work. Connecting these parts of the process and enabling the same level of configuration management rigor and control for product variability by utilizing the engineering product definition enables you to manage the entire product definition consistently and in coordination.
Increasing expectation of product variability	Your customers expect and demand not just a product that meets their functional needs, but one that is tailored to style, color, special features and conveniences of choice. Managing this range of offers efficiently as part of your product definition is a must. You can achieve it by: • Predicting and offering the variety that your customers demand with specific, carefully targeted product variants. • Allowing your customers more flexibility in the features they select.
The challenge of choice	Managing a distinct product data for each variant of a product may be feasible if you have a small number of product variants. However, as the product evolves, you may find that you are spending an increased amount of time managing and updating many configurations.
Leveraging the power of commonality	Across all variants for a given type of product, there is typically a large amount of data common across many or all variants. Harnessing this knowledge to formally manage this variability directly within your product data allows you to efficiently define and manage your product over its lifetime.
Product architecture drives products across platforms and generations	As products evolve and new generations of products are introduced, you can tap into and reuse your product knowledge by managing a template of product breakdowns. That template describes the functions, systems, logical elements or other breakdowns that are common across products or generations of products. This establishes a framework for reuse, product planning, and accountability for completeness of product definition relative to the expected content and variability. In addition, it enables you to formalize a corporate standard for how products are organized, planned, and delivered to support a repeatable measure of completeness and maturity as a product definition evolves.
Formally manage your product portfolio	You can introduce your product decisions by defining and marketing your product suite to your customers. These decisions flow from product planning through to engineering and production. It is important to have a formal representation of this product breakdown available to guide and account for work throughout the process. Reuse and applicability allow you to manage variability across your product portfolio and to guide what your customers are exposed to, what is expected and allowed to leverage. • Reuse to manage a corporate set of features that can be selectively leveraged as new products and new product generations are introduced.

	• Condition to ensure that within a broad library of features that are offered across the entire product line, there is a limited set that is relevant and allowed for a given product.
Engineering on demand	Even after carefully defining and presenting the variability that you wish to offer to your customers, you may allow them to configure orders for which not all of the engineering definition is complete. In this case, typically for specialized products, you can offer a range of dimensional variability and geometric or engineering dependencies. The variability intelligence you manage includes these engineering heuristics. This allows you to *Engineer to Order* some percentage of the parts for an ordered configuration on demand after receiving the order.
Voice of the customer drives product decisions	Ultimately, product decisions are made based on what your customers demand and will pay for to make the product successful and profitable. Often, there is a disconnect between capturing customer demands and how these get evaluated and approved in the product planning cycle from how this gets communicated and incorporated within the engineering product definition. Product managers, product planners, and the engineers and designers must be able to seamlessly handoff and share their work, to perform impact analysis, and be accountable throughout this process.
Separating product variability from product engineering	Variability is recorded against the product content. It is tied to the engineering definition of the product. That is, features and configurator rules are defined using nodes in your product as the context. This leaves little autonomy for the product managers who are doing early product planning to reflect the voice of the customer and determining what features and products to offer. Product managers need the ability to formally consider different features. They need to review, evaluate, whittle down, and, finally, decide what features should be offered to the market. These decisions have to be made before engineering needs to get involved, and without the need to expose product management to the engineering definition of the product.

Benefits of using Product Configurator

Product Configurator allows you to manage variability and to ensure consistency when authoring and qualifying your product data. You can define and manage variability for a product from the inception of requirements setting to the conclusion when an order is delivered to a customer.

Product Configurator is a single, variant backbone across full product definition lifecycle within Teamcenter. That is, the variability you define within Product Configurator can be seamlessly used to configure data ranging from SDPD to documentation, design, part master, manufacturing processes and operations, and ultimately even manufacturing execution. Benefits of using Product Configurator reach far beyond engineering.

Role	Tools	Activities	Benefits
Systems Engineer	• Logical features • Requirements	• Configure system breakdown.	• Ensure system variability and requirements

Role	Tools	Activities	Benefits
		• Configure logical blocks.	compliance accountability.
Program Manager	• Product line breakdown • Program targets	• Track progress to program targets.	• Ensure program decisions and targets are tracked and accounted for through product development process.
Marketing/Product Manager	• Market-facing product models • Market-facing (sales) features • Marketing constraints	• Configure market facing product variants. • Develop marketing materials to describe and promote product features.	• Provide product planning input to engineering and design.
Engineer/Designer	• Engineering and technical features • Map sales to technical features • Technical constraints	• Engineering configuration and analysis, such as tracking vehicles, overlay, and clearance. • Identify and analyze dependent configurations.	• Design, clearance for a range of configurations, at once. • Identify if any parts are missing or incomplete for any variant. • Directly consume and react to marketing inputs.
Documentation Specialist (future implementation)	• Features mapped to user manual content • Documentation features	• Configure, publish user, technical and service manuals.	• Automate inclusion of only relevant product documentation.
Manufacturing Engineer/Process Planner	• Manufacturing and process features • Mapping of engineering to manufacturing features	• Configure and define process based on engineering and customer variants. • Optimize and balance assembly line based on volumes.	• Directly leverage engineering variability and configurations to ensure process plan consistency without duplication.

Role	Tools	Activities	Benefits
	• Manufacturing constraints		
Customer/Dealer		• Consider available product features. • Configure and visualize customer orders guided by configurator wizard.	• Configure a product with the features customers want. • Visualize customer orders as they make decisions. • Ensure customers do not see choices that are not relevant to their orders.

3. Basic concepts for using Product Configurator

Basic concepts for using Product Configurator

The Product Configurator allows you to specify the variant data for a product at the concept stage before any detailed design is done. The variant data is independent of any application domain, for example, CAD design or part planning.

You create families (for example, **color**) and allowed features of those families (for example, **red** and **blue**). You then define a variant condition (for example, **only load IF color = red** is specified in the variant rule) on that content that is conditionally valid for a configuration based on selected features.

The variant data is collected within an item revision that represents the product lines which is called a configurator context. For example, you can associate the configurator context which collects families, features and configurator rules with the top level product design. You can then define variant conditions on partitions and design components.

To configure a particular variant of an assembly or product, select and apply an adhoc set of feature selections or a saved variant rule (a group of families and features such as **color = red**, **material = cotton**). A saved variant rule is stored in the database and retrieved later.

To suggest initial values or specify features or combinations that are *not* allowed, you can create configurator rules (for example, an exclusive configurator rule may specify to **error if color = green AND material = cotton**).

Such exclusions and inclusions are referred to as *constraints*.

The *variability* of an item revision is defined as a complete set of families, each with a set of allowed features (for example, **color = blue, silver, white**). You specifically associate this variability with the product. *Variance* is defined as the feature combinations that are available for a scope of the product.

CAD designers then create design solutions that satisfy the variance criteria and associate them to the appropriate design assembly or module, selecting the applicable variance. You can apply different variant configurations to the design data for procedures such as digital mockup analysis or interference checking.

Objects and data you work with

You should understand the purpose of the following objects and data you may manage in Product Configurator.

100% BOM	The “as sold” product configuration, for example, the configuration of a car that will be built and shipped to the dealer.
120% BOM	An overlay of multiple combinations that would not occur together in a buildable configuration of a product but are brought together for evaluation or analysis across a range of possible configurations. You cannot build the product from a 120% BOM.
150% BOM	A combination of all possible variant configurations that could not be brought together in a buildable product but represent a full range of content for all possible variations. You cannot build the product from a 150% BOM.
Allocation	The use of features, families, and groups in a specified context. Features may be shared across multiple configurator contexts or dictionaries.
Allocation source	The source of a configurator allocation is the configurator context or dictionary from which the object is selected.
Allocation target	The target of a configurator allocation is the configurator context or dictionary to which the object is allocated.
Condition	An optional statement for any configurator rule that defines preconditions for which this rule will be relevant. If no condition statement is specified, the given configurator rule is valid for any configuration for a given configurator context. If there is an applicability specified, the rule is not evaluated if the applicability statement is not met.
Availability	Specifies that features are designated as valid within a configurator context. For a *positive availability* bias, all features allocated to a configurator context are implicitly available for all product models within that configurator context. For a *negative availability* bias, all features allocated to a configurator context are by default unavailable for any of the product models within that configurator context. You must explicitly make each feature available.
Application model	A different application model container, such as product design or partition template, to hold data that models specific aspects of a product. For example, the engineering BOM may be managed in a different application model than the manufacturing BOM.
Architecture template	Architecture templates that define the partition setup. The template can extend many levels deep and define your vehicle. A partition breakdown can be saved as an application model and can be called an architecture template. Once that template is created, it can be reused.
Availability Bias	Characteristic of a configurator context. Availability can be positively or negatively biased.
Conditional value default	
	A logical condition added to a default feature so that the feature only defaults under certain conditions. For example, set the **color** feature to **red** only if the **material** feature is set to **cotton**, but you may still make a choice other than **red** for the **color** feature. Teamcenter evaluates all of the constraints simultaneously. Therefore, if default rules conflict with other constraint rules,

	then defaults are dropped and ignored during the configuration expansion process.
Configuration context	A set of recipes or filters that are applied to the product data to configure the desired variant. These recipes may include maturity, family selections, and effectivity.
Configurator context	An item that defines the scope of the families and features for a product. A configurator context may represent the product, product line, business unit, or operating unit in which features are allocated and rules are authored. It collects the features and configurator rules relevant for a given product line. Configurator rules may apply to several configurator contexts (that is, to product models across contexts), but do not refer directly to the configurator context. There must be at least one configurator context in the system, but you may manage more configurator contexts to suit your business needs.
Configurator rule	A Boolean expression recognized by the Product Configurator. It represents a statement to evaluate, specify, restrict, or include allowable feature combinations or model applicability. A configurator rule is a variant expression that sets one or more features if any stated preconditions are satisfied. It consists of optional Boolean preconditions and a Boolean statement that specifies a value to set for one or more features. *Exclusion* rule is a variant expression that disallows certain combination of features to be selected together under specified conditions. If the constraint expression is satisfied, the rule fails. *Inclusion* rule is a variant expression which forces certain combination of features to be selected together under specified conditions. If the constraint expression is not satisfied, the rule fails. *Availability* rule is a variant expression that specifies which allocated features are allowed in a configurator context. *Exception* rule is a variant expression that defines exception conditions for a dynamic family. Note: This rule is only available if your administrator enabled the creation of dynamic families. A *Free-form* rule allows you to create more complex free-form SMT-based rules using the SMT Lib scripting language. Using free-form rules, you can express restrictions and constraints that are not limited to a Teamcenter expression format for a standard configurator rule. Configurator rules are workspace objects with their own names, description, and owner. The associated error messages may be localized.
Default feature	A feature that is preset for a family by the system before you actively make any configuration selections.

Dictionary	A collection of groups, families, and features, making them available to share and reuse.
Family	A method of grouping features with a similar purpose. For example, you can create a family called **color** with features of **red**, **green**, and **yellow**. A family may be allocated to one or more dictionaries or configurator contexts, and family memberships may change over time. A selection from a family may be optional (allowed but not required) or mandatory (required) for a complete, valid order string.
Group	Used to organize families for view and navigation, as well as used to lead you through a guided interactive configuration. A family may belong to more than one group and its group memberships may change over time. All features within the family are included in the group, but the group to family relationship is not an allocation or means of filtering features. Groups are associated with at least one configurator context or dictionary.
Feature	A feature represents a possible value in a family. Features are workspace objects with their own names, description, and owner. A feature belongs to one, and only one, family. Features may be free-form or a member of an enumerated list.
Free-form family	A family that defines an allowed range for the user input including one or two Boolean operators. During order string creation, the user or system may specify a value for the free-form family.
Global rules	Rules that apply across all product contexts and apply irrespectively of the model. You can reuse and share global rules across multiple products.
Intent	An intent is the purpose or reason why an object was authored or used.
Mandatory	A selection must be made to have a complete, valid configuration.
Marketing feature	A feature that is valid for exposure to the Product Configurator user and is applicable globally. It is defined independently of product data, but may be referenced directly by product data or may derive one or more technical features that are referenced within the product. Note: A formal distinction between engineering and marketing features is not supported in this version of the Product Configurator.
Model Applicability	An optional statement for any configurator rule that defines one or more product models for which this rule will be relevant. If no model is specified, the given configurator rule is valid for all models within the configurator context. If there is a model applicability specified, the rule is not evaluated if any other product model is selected.
Multiselection family	A family for which more than one feature may be selected for an order string.
Mutual exclusivity	Specifies when a set of possible selections may only have a maximum of one selection for a valid configuration. Mutual exclusivity often refers to the features of a family, that is, only one feature may be selected for a given family.

	However, it may also apply more broadly, for example, if you select a feature for one family, the system then disallows selection from one or more other families.
Namespace	Ensures all families and features in the system are sufficiently unique that the Product Configurator rules solves inherit their namespace from their families.
Package	A combination of individual feature packages grouped together to form a package. Member features may come from different families. Package features are set automatically when you select a package. Package features are connected using a logical AND operation in the variant expression. *Packages* allow you to define a business-relevant collection of features that should be selected together based on selection of the package family itself.
Partition	A partition is a container that helps you organize and find data. The data can be organized in multiple ways, such as by function, spatial location, or physical description. Partitions can be used to divide a large product into smaller, more manageable sections.
Product line	A product line structures product models into a hierarchical tree based on common business characteristics. Product lines are used to group related product models. You can look at a product model as a child of a product line. You can also define a hierarchy within a product line where a product line breaks down into other product lines. Eventually, the leaf level product line breaks down into product models.
Product model	A designated product or subproduct that represents a market-relevant product offering, for example, a vehicle or a major assembly such as an engine or transmission. A product model may only be created in a configurator context. You can reuse families, features, and product data (for example, BVR structure, partition template, and design components) for more than one product model. The product model may be defined by a combination of features; in this case, the product model is referred to as a *package model*. *Package models* allow defining of what features necessarily comprise the base definition of the market-facing product model; these features automatically get selected upon selection of the product model itself.
Saved variant rule (SVR)	An expression that enumerates a selection of features that may represent a complete or partial configuration. A variant rule must be validated as complete and consistent with no violations to produce a buildable physical variant.
Standalone feature	Standalone features are arranged into dynamic families with the intent to establish mutually exclusive behavior. Standalone features are optional features. A standalone feature may be a member of one or more dynamic families concurrently. Your administrator can configure the **Cfg0EnableDynamicFamilySupport** preference to enable creation of dynamic families and standalone features. Note: The standalone features and the dynamic family concept are in an incubation phase. They should only be used with the approval

	from Siemens product management. They require authoring in Active Workspace and consumption of configurator data.
Summary	A collection of feature summaries which allows you to summarize the features from the single family in the configurator context or in the dictionary. A feature summary allows you to efficiently author rules by referring to the *summary* when the rule applies to all of the features it summarizes. A summary feature may be referenced in variant, exclusion and inclusion rule expressions. For example, you can create a summary for all V6 engines which have individual feature summaries as values. A feature summary is not visible during configuration.
Summary model	A collection of models that allows you to summarize the product models in the configurator context. A summary product model allows efficient reference to any summarized product model in a single statement. A summary model behaves similarly to a summary. Summary model features are connected using a logical OR operation in the variant expression. Summary models may not be nested. Product model members must be from the same configurator context. Product models may reference one or more features. A summary model family may be referenced in variant, exclusion, and inclusion rule expressions.
Variant	A specific configuration of the generic product that removes ambiguity and represents a physical product or subproduct that could be manufactured. A variant may not represent a configuration of the full product but, for example, may refer to a single feature combination satisfied for a partition.
Variant configuration	Allows you to validate a configuration against configurator rules such as availability, inclusion, and exclusion rules. When you configure and assign features to families, Teamcenter may evaluate the constraints and return a list of information, warnings, and errors on the various combinations of the families.
Variant condition	A rule that controls when an element of the product data is included in the configuration. If the rule evaluates to true, the qualified product data element is included in the configuration.
Variant criteria	Variant criteria is a revisable subtype of a variant rule. You can release, status, and revise variant criteria.
Variant data	Variant groups, families, and features.
Variance	The permutations of features offered for a specific design solution or generic product.
Variant expression	A general term that refers to any variant rule. It may refer to one or more product models, summary or package product models that span multiple configurator contexts.

What is a dictionary?

A feature dictionary collects variability of your entire product suite, such as:

- Collects groups, families, and features, making them available to share and reuse.
- Allows you to formally group or create subsets of groups, families, and features to reuse between specific products or product lines.
- Allows you to share and reuse features. You allocate from one or more dictionaries or **configurator contexts** to formally create subsets of the features that is relevant to one or more configurator contexts or other dictionaries.
- Provides a natural way to segregate data relevant to each user communities, for example, by business unit.

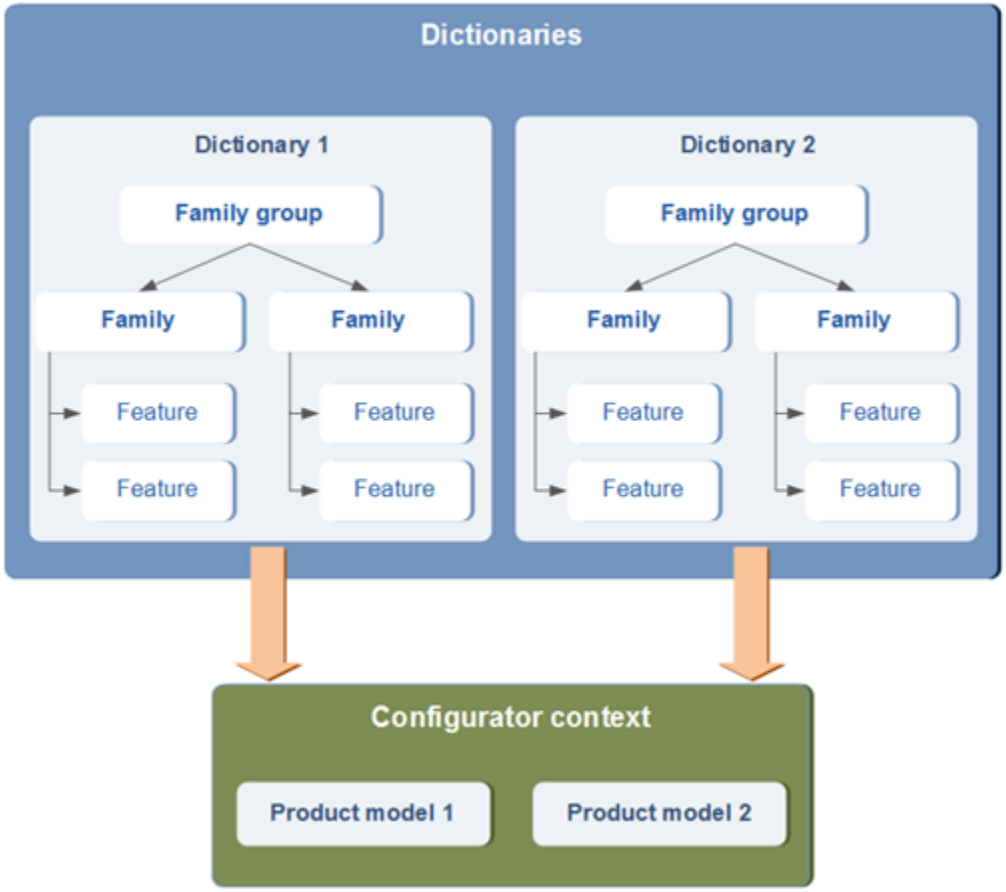
You can create multiple dictionaries, but the use of dictionaries is not compulsory. You can create different dictionaries for different functional responsibilities. For example, an automotive company may need a *vehicle* dictionary and a *powertrain* dictionary.

The decision to create dictionaries is based on your company's requirements for scoping and ownership. If your company chooses not to leverage a separate dictionary, you can create variant data directly within a configurator context for a given product line. You can reuse features by **allocating** them directly from one configurator context to another.

The dictionary manages families. Family objects have a family name, a description, and a reference to the dictionary item with the **parent_item** attribute. The dictionary defines a namespace in which families have a defined meaning. For example, the family **CAPACITY** may exist for both refrigerators and washing machines. The two **CAPACITY** families should exist in the context of different dictionary items as follows:

- The family **CAPACITY** when measured in liters references a dictionary item representing refrigerators.
- The family **CAPACITY** when measured in kilograms references a dictionary item representing washing machines.

The dictionary may define characteristics and boundaries that apply to all features in this family across all possible uses. For example, you can define the **COUNT** family as an INTEGER type family, whose value must always be greater than or equal to zero.



Note:

Your administrator can control who has access to create and allocate objects to the group, family, and features in your dictionary.

You can also work with *data dictionaries* in Teamcenter. Data dictionaries are central repositories for key building blocks or components of designs used in typical functional and logical design activities. These repositories make it easy to reuse components in multiple designs across projects or programs. Data dictionaries are modeled as libraries in the Classification application and can be associated with projects or programs, which allow designers who are project members to add data from the dictionary to functional, logical, or physical model structures.

What is a configurator context?

A configurator context represents a family of products that may share the same set of groups, families, and features.

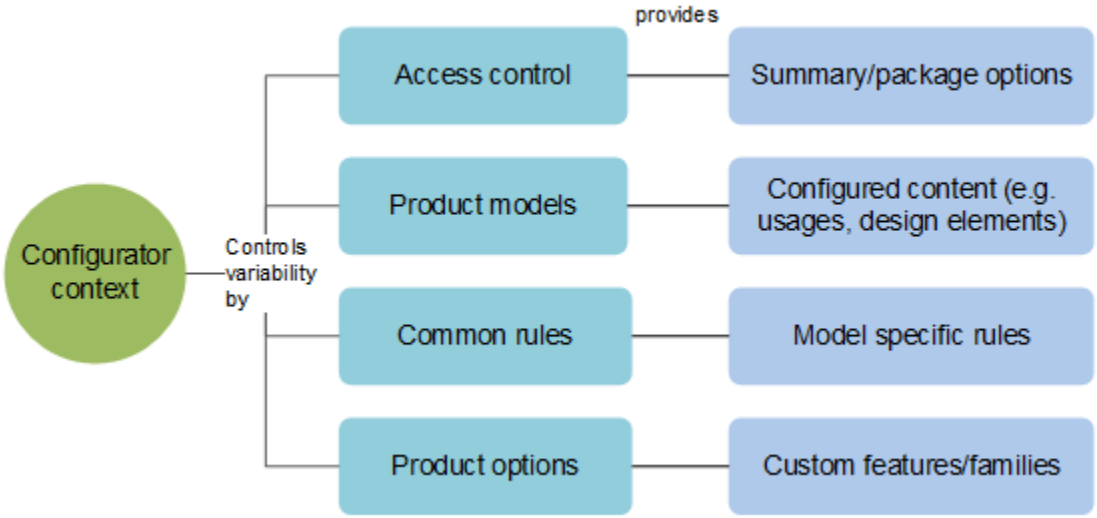
For example, cars and vans may be two different product lines that can be represented by two separate configurator contexts.

In addition, it allows you to define product models and reference configurator rules. Rules you create with the Product Configurator can apply to product models across several configurator contexts. A configurator context may be shared by multiple domains like design, engineering, process planning and others.

A product model represents a specific market-facing product to be offered. Specific features from the product line can be made available for each product model.

For example, a product line may have gasoline standard, gasoline luxury, diesel standard, and diesel luxury models.

Teamcenter does not *require* you to create configurator contexts or a dictionary. If your reuse requirements are not complex, you can create a single configurator context and minimize user interactions with it.



A configurator context:

- Defines and encapsulates one or more product models.
- Accepts the allocation of features that you can make available to one or more product models.
- Allows any rule to refer to multiple product models within or across configurator context items.
- Has no limitations of scope or scalability; the scope is determined dynamically for each product model.
- Provides a natural way to segregate data relevant to each user communities, for example, by product line.
- Ensures that sharing of features, configuration rules, and product content is limited to appropriate groups and products. Users typically work primarily in a single configurator context.

- Provides a boundary for enforcement and validation of business rules.
- Enables different business units to operate with a positive or negative bias for feature availability, as appropriate for their defined rules.
- Enables autonomous business units or component groups to share features and rule data when relevant.

Understanding allocation and availability

The term *allocation* refers to the use of variant data, such as features, families, and groups in a specified configurator context. Features may be shared across multiple configurator contexts and dictionaries. A feature is automatically allocated to the configurator context or contexts selected when it is created.

Configurator allocation is a relation between configurator context and variant data. Configurator allocation is a workspace object, and it can have its own life cycle. Configurator allocation can be configured using effectivity. It can also have multiple revisions. You can view configurator allocation objects in the **Variability Explorer** view.

You create availability rules to define availability, for example, to specify features that are allowed for one or more product models or summary models.

Authoring the availability expressions for a given **Subject** allows you to reduce the administrative overhead. Multiple conditions for a given subject can be authored, reviewed, and released in a single rule, such as a single availability rule can make a feature available for many product models.

For example, you may have six engines in your dictionary for the entire product suite. Out of those engines, three engines are relevant for a specific passenger car product line. You can choose to make only two engines available for the Anaconda EX product model within that passenger car product line.

> Note:
>
> If you plan to use revision rules to configure only released features revisions, you must also release the configurator allocation and family objects.

Configurator allocation follows WYSIWYG (what you see is what you get) behavior. What you select is exactly what is allocated.

Select	Configurator allocation behavior
Feature	If its family is already allocated to the target, then it is allocated because a feature cannot exist alone in a configurator context.
Family or group	Allocated as it can exist on its own.

Note:

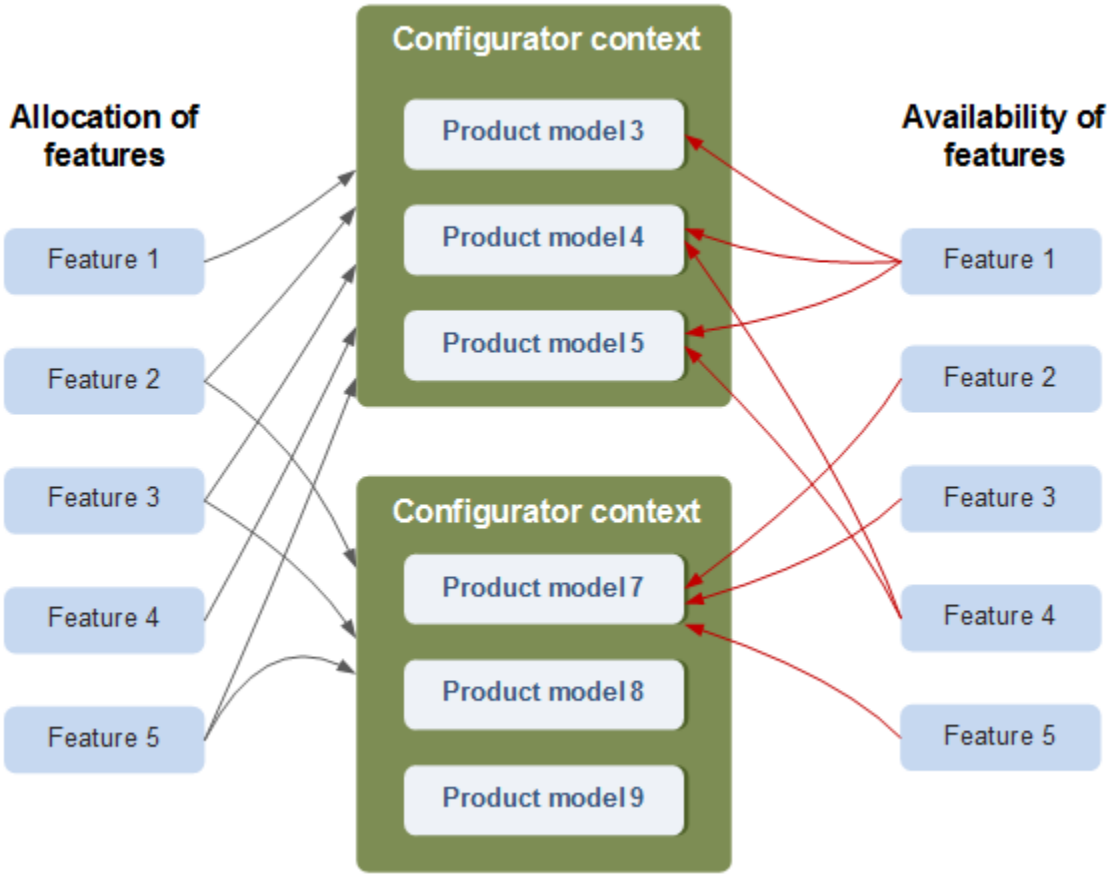
You can allocate a Boolean family independent of its Boolean value. However, it is not recommended to treat a Boolean family differently from its value.

Packages must be allocated to a configurator product context and made available to each allowed product model.

Summaries are immune to availability. However, they have to be allocated to the context if you plan to use them in that context. Summaries are always available. However, the members of a summary are subject to availability.

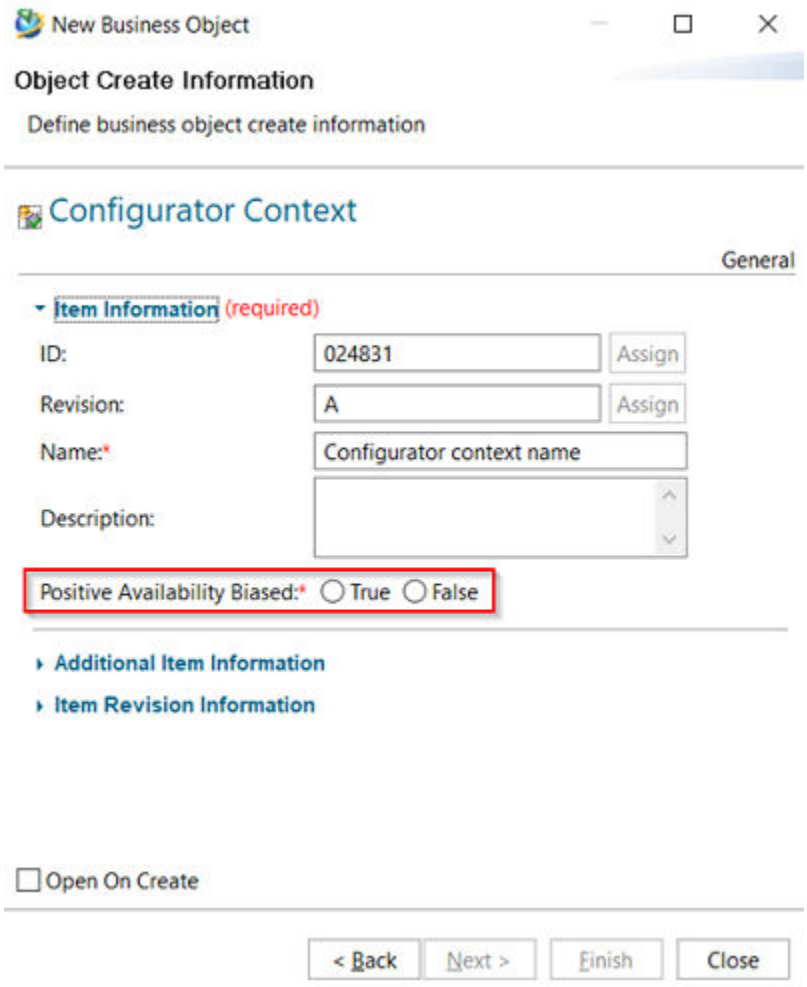
The *source* of a configurator allocation action is the configurator context or dictionary from which the object is selected. The *target* of the allocation is the configurator context or dictionary to which the object is allocated.

Once allocated, you can additionally specify what set of the features allocated to the configurator context should be available for one or more product models, product lines, and summary models.



Understanding availability bias

You can specify availability bias for configurator contexts using the **New Business Object** panel.



Availability can be positively or negatively biased. It cannot be changed later. This can be specified per configurator context.

Positive availability bias

All features allocated to a configurator context are implicitly available for all product models within that configurator context. No explicit action beyond allocation is needed to make the features available.

A positive bias minimizes the administrative overhead when introducing new features. A new feature is automatically available as soon as it is allocated to a configurator context.

- Allocation of any feature to a configurator context makes the feature implicitly available for all product models within that configurator context.

- As new product models are subsequently created, all allocated features are automatically available for the new product models.
- To restrict a feature, an administrator must create exclusion rules to state explicit exceptions for restricted features.

Negative availability bias

All features allocated to a configurator context are by default unavailable for any product models within that configurator context. You must explicitly make each feature available using availability rules. This can be done using the **Availability Matrix**.

A negative bias minimizes unexpected side effects when introducing new features. A new feature is not available unless an administrator explicitly approves its availability rule.

A negative bias can be used to reduce the variability displayed to users after selecting a model. As the number of features are reduced, performance can be optimized for models.

Make sure that every mandatory family has at least one feature available for every model. Otherwise, the configurator will not find a single valid combination for that model.

An appropriate license level is required to create a negatively biased configurator context. If the license is not available, the system generates an error during the creation of the negatively biased configurator context.

Configurator rules

Configurator rules or constraints allow you to specify what features and feature combinations are allowed for a given product for a valid configuration or customer order.

The configurator rules not only guide internal product development, but also guide the end customers to ensure they are making compatible and complete selections. Feedback from applying the configurator rules during configuration enables you to ensure that you are making valid choices.

For example, during configuration, a customer makes only one selection: a product model. Then, the configurator rules guide the customer to specify valid choices and make compatible selections.

	Availability rule	Specifies that a feature is allowed and available. For example, only available features are shown when configuring your configurator context and selecting a product model.
	Default rule	In the case that a selection is not made by a user or by an inclusion rule during configuration, the system sets

		the specified feature. Once the rule is set, it may be overridden by the user.
	Inclusion rule	When the specified features are selected, additional features are automatically selected for the configuration.
	Exclusion rule	When the specified features are selected, additional features are automatically excluded from the configuration.
	Exception rule	Defines exception conditions for a dynamic family. Note: This rule is only available if your administrator enabled the creation of dynamic families and standalone features.
	Free-form rule	Allows you to bundle multiple constraints under a single revisable WSO with one owner, one approval process, one responsibility, and one effectivity when it has the same severity level. These rules are created using the SMT Lib scripting language.

All of the interactions with the configurator rules are grid-based, table-based, or in the form of text for free-form rules. This makes the selection of features intuitive and allows you to easily work with large sets of data.

Constructing configurator rules

Teamcenter supports several types of configurator rules, but each type comprises two components:

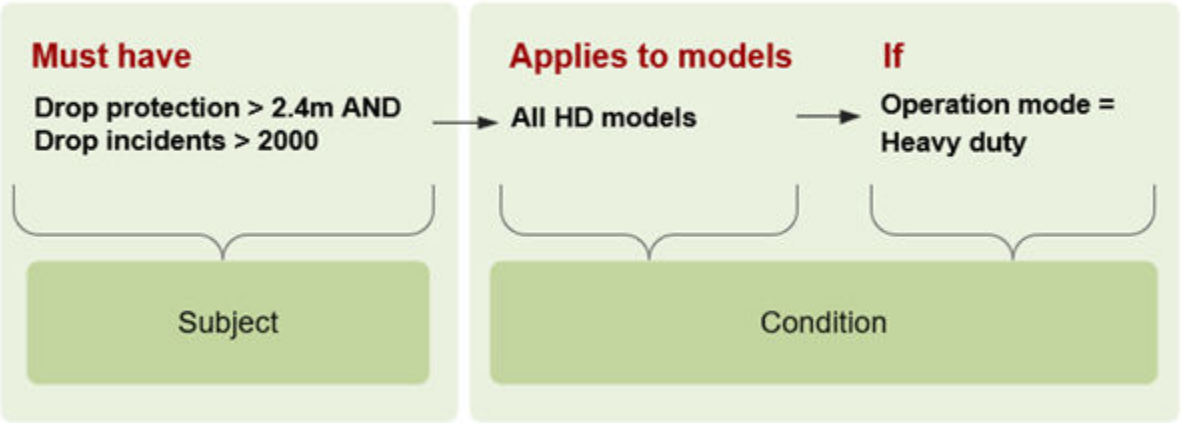
- **Subject**

  Specifies a feature to be made available, included, excluded, or used as a default selection.

- **Condition**

  Determines conditions under which the rule is applied.

  The **Subject** of the rule is applied to the extent the input criteria satisfy the **Condition** condition of the rule. **Condition** can consist of product lines, models, and regular features.



In this example, for a given product there should be a drop protection of at least 2.4 meters and for at least 2,000 incidents. The rule is enforced for all HD models when the specific operation mode is set to heavy duty.

In addition to configurator rules, you can create more complex free-form SMT-based rules using the SMT Lib scripting language. Similar to standard configurator rules, free-form rules can be split and revised and can have effectivity and severity.

Understanding the configurator namespace

The solve engine requires a minimum level of family and feature uniqueness to unambiguously run a validation or solve an order. The namespace provides the minimum level through a property on a family. Users select a namespace from a list of values defined by the administrator.

The role of the namespace includes the following:

- Acts as a boundary for the uniqueness of a family or feature.
- Is a user-selectable property on a family that applies to all the associated features.
- Has no impact on scoping of variability or configuration, other than controlling uniqueness.
- Has no inherent correlation or significance relative to the configurator context for which a feature is created.

The uniqueness of the default namespace depends on the level of uniqueness needed by the solve engine to always unambiguously understand which features are considered for validation and solve. The solve engine must always pass a feature string that unambiguously identifies each feature.

In some instances, the uniqueness provided by the namespace is insufficient for unambiguous validation: a greater threshold is required. For example, sometimes Teamcenter imposes a level of uniqueness on family and feature data through the use of namespace that cannot unambiguously run a validation or order solve. The order strings from an external sales configurator or other external system

cannot unambiguously correlate each feature in the order string to a Teamcenter feature revision. In this case, you must define a uniqueness threshold to ensure that the incoming order string information can always be correlated to individual feature revisions in the database.

If a greater threshold of uniqueness is needed, you can use multifield keys. That is, you specify a combination of properties that can be enforced as unique across all business objects. In this case, the uniqueness key for an object includes (at a minimum) the UID.

You can use multifield keys to enforce features that are globally more unique than the solver requires. This ensures the incoming feature string does not require the sales configurator to know more than the feature name to provide the solve engine with sufficient information to perform the solve. For example, you may use multifield keys to differentiate between the same feature name used by different organizations in your business — **Blue + Truck Division** is not the same feature as **Blue + Car Division**. In this example, **Truck Division** and **Car Division** are defined as multifield keys.

For example, the following scenarios use a fictitious Innovative Motors Inc. company. Innovative Motors Inc. uses Teamcenter with the default setting.

- Innovative Motors Inc. creates a family **Interior Color** with the **Red**, **Green**, and **Blue** features. The **Red** feature is not used in any other family. Product orders may refer to **Red** as an unambiguous characteristic of the product.

- Innovative Motors Inc. creates another family **External Color** with **Red**, **Black**, and **White** features. The **Red** feature is no longer unique. Product orders may no longer refer to **Red** as an unambiguous characteristic of the product. They are forced to refer to **External Color = Red**.

- Innovative Motors Inc. acquires company Future Car Ltd. The products of Future Car Ltd. and Innovative Motors both use families **Interior Color**, but their meaning is slightly different. For the combined enterprise, the family **Interior Color** is no longer unique. Product orders may no longer refer to **Interior Color = Red** as an unambiguous characteristic of the product. They are forced to refer to **[Innovative Motors Inc.]Interior Color = Red**.

A feature must be unique for the entire organization and in a configurator context. A feature name must be unique for a family. Product model name must be unique in a configurator context. Additionally, the combination of a family ID and a family namespace must be unique in a configurator context.

> Note:
>
> By default, a family is unique in a namespace and a feature is unique in a family.

In both Teamcenter clients, the following values of the default **Cfg0DefaultOptionFamilyNamespace** family namespace preference are honored while authoring a family under a context or a dictionary:

- The default value is **Teamcenter** and it can be any string value to make that family unique.

- If you want the namespace to be any string property of the configurator context or configurator dictionary, set the value in the preference as {item_id} or {object_name}.

- Currently the default value is **Teamcenter**. You can modify this to any other string value like **Test**. While authoring in Teamcenter clients, you can also modify the family namespace value. You are allowed to remove the family namespace that is already populated. Then the system saves the family with an empty namespace. However, it is not a recommended option to have an empty namespace.

- You can change the default namespace value populated on the family row to a desired value at runtime while authoring the family.

  The family namespace is an editable field before saving the data. It cannot be modified after saving the family.

- The system always defaults the model families to the context ID and this is not configurable.

- If the family is being created independently without the context and without client, then the **Cfg0DefaultOptionFamilyNamespace** preference is not honored.

4. Defining product configurations

Creating variable products

Creating variable products

Managing a discrete product variant for every possible family combination is inefficient and error prone. Instead, you can choose to leverage the commonality across the product and build the variability into the product data. You manage the variable product definition and derive all valid product variants from it.

Once you have identified the commonality in the generic product, you define:

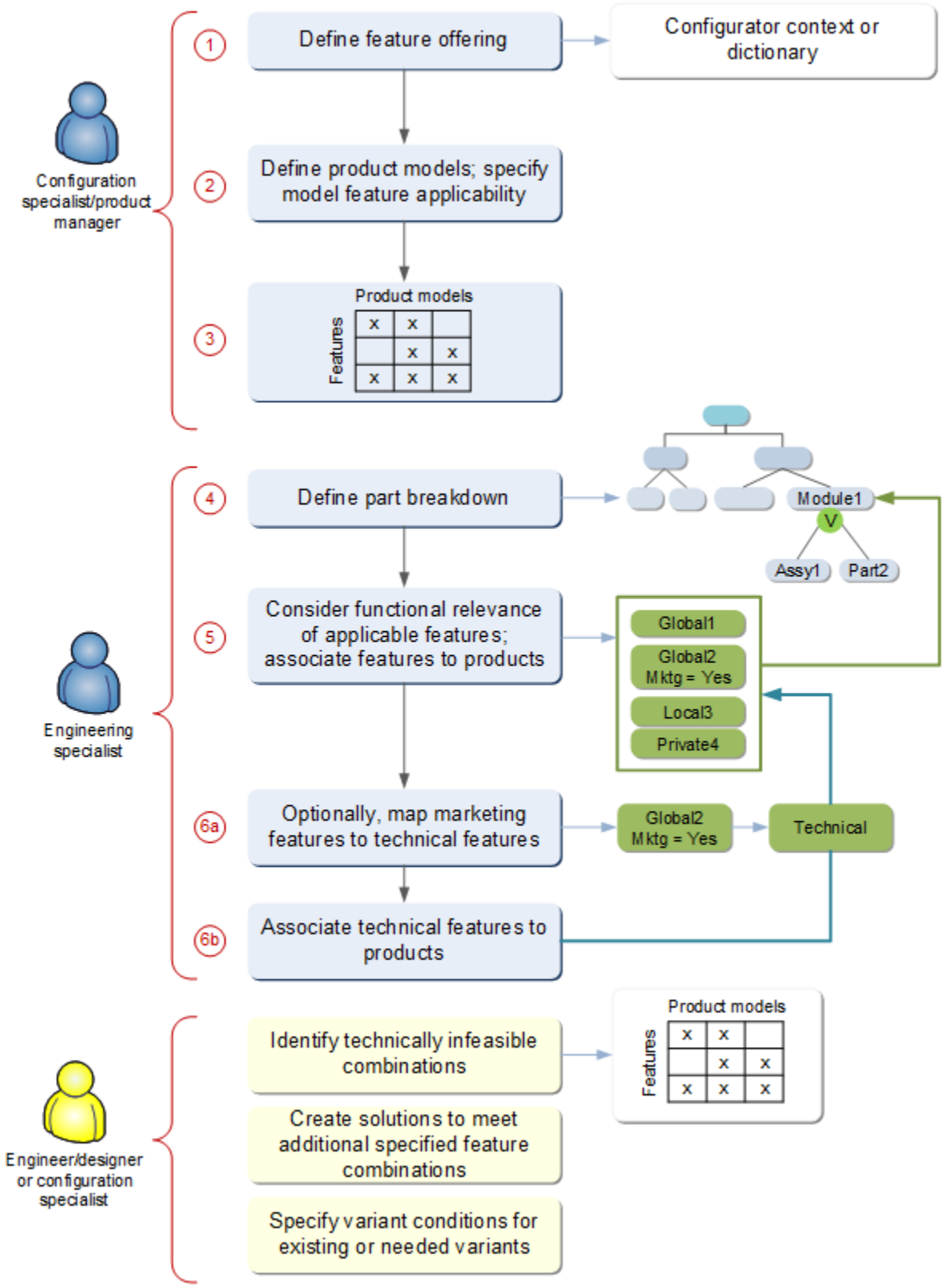
- The valid feature combinations for that generic product.
- The variant rules and constraints needed to ensure that the valid combinations are allowed but no others.
- The design or part solutions required to satisfy any variant within the range of valid variants.

You perform these steps when creating the variable product definition, regardless of whether you manage a conventional product decomposition or specify variability relative to a 4GD partition breakdown.

You may not always perform these steps in the same order each time. You can always see the valid feature combinations specified for a generic product, and also the design solutions specified for the generic product. This is possible even if the solution or the feature combination is not specified, or if they are specified but not yet linked.

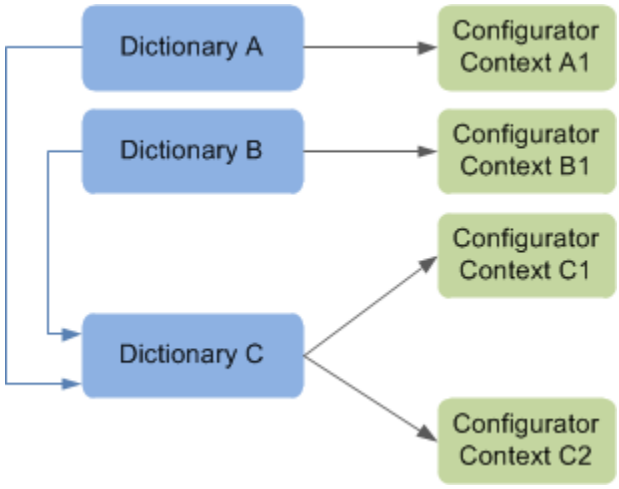
> Note:
>
> If you define variability against 4GD partitions, it may not be possible to specify a solution except when it is linked to a particular set of features.

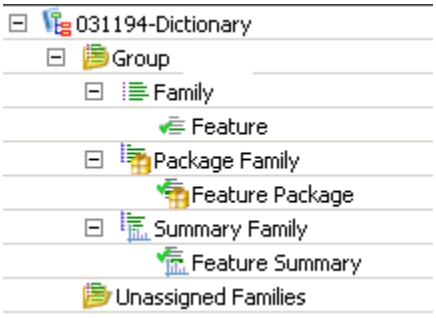
The following diagram shows the task flow to create a variable product.

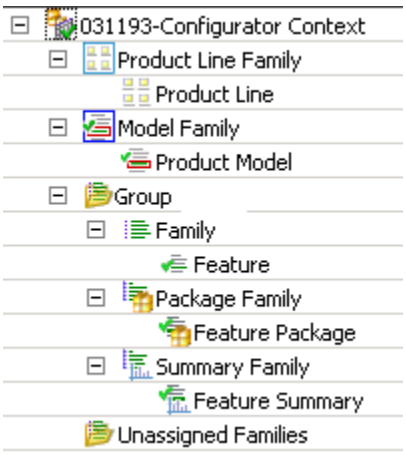


Managing variant data for products and dictionaries

You can create **configurator contexts** or *dictionaries*, and then associate feature families and their features with them. This allows you to reuse variability across multiple product lines and share features across multiple specialized dictionaries. Once created, you can add custom images to these configuration objects to make them easily identifiable. You can create constraints (including defaults, availability, inclusive, or exclusive rules) and associate them with the configurator context or dictionary. You can use Teamcenter to manage the life cycle of the configurator context or dictionary, for example, by using workflows to approve additions to them.



Role	Responsibility
Configurator expert, configurator administrator, or configuration analyst	Create variability in dictionary. 
Product line leader	Allocate variability from dictionary to configurator context.

Role	Responsibility
	

Note:

If you create a new configurator context or dictionary by copying an existing configurator context or dictionary (that is, using the **Save As** command), variant data associated with the existing configurator context or dictionary is not carried forward.

Updating and managing variable product data

Once you have created a variable product definition, it may not remain static for long. Any of the pieces of product data within the variable product definition may evolve, including saved variant rules, default rules, and variant rules for compatibility. The changes in the product data must be folded into the variable product definition. To do this, you must implement configuration control of the prechange configuration to allow a controlled introduction of the changes.

You may encounter this situation if product or marketing managers update the features offered for a product model. They may define features that will now become available, will become unavailable (obsolete), or will be available only in a combination not previously allowed.

If you defined allowed variability directly against a product, you follow these steps to reflect the changes in the product data:

1. Engineering evaluates feature availability changes and determines if new solutions are required to satisfy the updated features or if existing solutions become obsolete as a result of the change.

2. Engineering determines if any new families or derived default rules that set features should change.

3. If the product impacted by the change is currently released or uneditable, Engineering typically revises the product and makes necessary adjustments to the revision. This is achieved by specifying effectivity on the revision to match the effectivity of the family changes. All variant conditions are effective based on the effectivity of the parent on which they are qualified.

4. Users ensure that saved variant rules, derived rules and variant constraints are effective and consistent with the product data update.

5. Family and feature updates are reviewed and released.

6. New and updated product data is reviewed and released.

If you defined allowed variability against a 4GD partition breakdown, follow these steps to reflect the changes in the product data:

1. A configuration specialist updates feature combinations that are relevant for a particular partition. Depending on your business process, you may revise the partition item to introduce the updated feature combinations and corresponding engineering solutions.

2. The partition is reviewed and released.

3. Engineering reviews the updated feature combinations and may certify that an existing solution is valid for an updated feature combination. Alternatively, they may identify and link a new solution that satisfies the feature conditions.

The change may be driven by a necessary design change that is not related to any change in family availability. If you defined the allowed variability directly against the product data, you follow these steps to reflect the changes in the product data:

1. Engineering makes a change to a design. If the design is not in an editable state, you revise the impacted design to make the change. You then update the engineering content of the design, for example, the CAD file.

2. Engineering evaluates all variant conditions applied against any updated node. Any variant conditions that have become invalid as a result of the module update must be updated and made valid. For any new nodes, a variant condition must be applied.

3. Engineering performs a solution replacement (replaces the design in the product) for any updated product data.

4. Engineering ensures all specified feature combinations have valid solutions identified.

> Note:
>
> This operation should not result in any new families or features relevant to the selected product node. It should not require updates to derived configurator rules or compatibility rules.

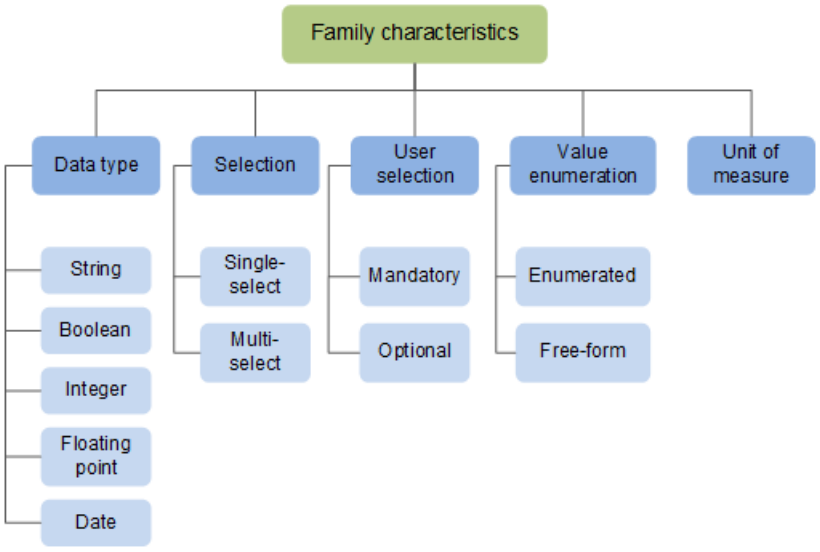
Working with families and features

Feature families characteristics

Families and features

Families and features are the basic objects of any variant scheme. You create families using terminology that users understand, for example, **color**. You then assign allowed features to those families, for example, **red** and **blue**. You associate families and their features with an item revision that represents the product library.

Families represent the configuration questions being asked; while features are the allowed answers to those questions.



The floating point and integer families can only have one unit of measure (**UOM**) defined. The **UOM** assignment is not mandatory.

For example, you specify **cm** for centimeters as the **UOM** type for your integer family. There can be one or more integer or floating point families with **cm** in the **UOM** type. You are not allowed to create values with **kg** for kilograms in the **UOM** for that integer family. Feature values such as **4**, **4cm**, or **4 cm** are considered identical, and creation of only one of these values is successful. If you create a feature with ID of **4cm**, then subsequent creation of values such as **4**, **4 cm** is not allowed.

Only the ID of an authored feature is allowed in the configurator expression. For example, if the feature is authored with the ID of **4cm**, then all expressions that reference that value must contain **4cm**. An expression with IDs of **4** or **4 cm** is not supported.

Note:

Multiselect families may cause significant overhead to the processing. Siemens Digital Industries Software recommends to avoid creating multiselect families when possible. In most cases, you can use Boolean families instead of multiselect families. Unless your company requires features to be explicitly managed under one family, the use of multiselect families is not recommended.

You can release and revise a family. Both a family and its allocation are revisable.

The following are examples of how the system configures revised families with features.

- You release and revise your family to change it from mandatory to optional or to change a min-max value. Depending on which revision is configured for a family in the **Variant Configuration** view, the system considers either an optional or a mandatory behavior for validation and completeness.

- After revising your family, you add a new feature to the new revision of the family.

  Only when the new revision of the family is configured, the system:

  - Displays the new feature in the Product Configurator views, such as, **Variability Explorer** view and **Variant Configuration** view.

  - Considers configuration rules that include that new feature during the validation in the **Variant Configuration** view.

Types and behaviors of families

Families may have the following behaviors specified:

Discretionary (optional)	Families where selection is optional. When you do not specify a feature during configuration, you are allowing Teamcenter to choose a feature for an optional family based on the system rules.
Mandatory	Families where an empty value ('') assignment causes a criteria validation violation (except for logical values) with the message associated with L18N key **k_variant_criteria_outside**, and UNSET means *any of*.
Free-form	Families, in contrast to fixed enumerated families. You cannot add or allocate fixed values to free-form families. Free-form families cannot be multiselection families.
Multiselect	Families, in contrast to single selection families whose features are mutually exclusive. *Multiselect* families cannot be free-form families.

You can also create numeric families. If you classify a numeric family as a discretionary or optional family, then you must also create a default for this family. Alternately, numeric families are mandatory. The same behavior applies to date families.

Once a family is created, there are several values, such as **Free-form**, **Multi-select**, and **UOM**, that are not modifiable. The system is designed to deny these updates to prevent expressions in rules and variant conditions from being evaluating in unintended ways. The following example illustrates the need for this restriction.

For example, you create a family with three values.

Mirror = {LeftMirror, RightMirror, CenterMirror}

You mark the family as single select during creation. Because it is marked as single select, all the values are mutually exclusive. Once the family is marked as single select, existing data such as rules and variant conditions build on this knowledge.

In this case, the expression **Mirror!=LeftMirror** means that the **Mirror** can be **RightMirror** or **CenterMirror**. Similarly, the expression **Mirror=LeftMirror** means that the **Mirror** can neither be **RightMirror** nor **CenterMirror**.

If the family is modified to be multi-select instead of single select, the meaning of the existing authored expressions in rules and variant conditions has changed.

Now, the expression **Mirror!=LeftMirror** does not mean that the **Mirror** can be **RightMirror** or **CenterMirror**; it simply says that the **Mirror** cannot be **LeftMirror**. There is no statement about whether or not the **RightMirror** or **CenterMirror** is there.

Similarly, the expression **Mirror=LeftMirror** does not mean that the **Mirror** can neither be **RightMirror** nor **CenterMirror**; it simply says that the **LeftMirror** is there. There is no statement about whether or not the **RightMirror** or **CenterMirror** is there.

Dynamic families

Your administrator can configure the **Cfg0EnableDynamicFamilySupport** preference to enable creation of dynamic families and standalone features.

> Note:
>
> The standalone features and the dynamic family concept are in an incubation phase. They should only be used with the approval from Siemens product management. They require authoring in Active Workspace and consumption of configurator data.

Dynamic families are similar to literal families, but dynamic in nature when grouping its features. A dynamic family can only group or organize features of the **Standalone Feature** type. In a dynamic family, the relationship between standalone features is loosely coupled, unlike in the literal feature families.

Music System	Dynamic Family
DVD	Standalone Feature
MP3	Standalone Feature
Radio	Standalone Feature
Speakers	Dynamic Family
4W	Standalone Feature
8W	Standalone Feature
12W	Standalone Feature
Safety Group	Group
Airbags	Dynamic Family
2	Standalone Feature
4	Standalone Feature
6	Standalone Feature

Family data types

Families are used to organize features by common characteristics. They are unique in the context of the item in which they are created. The combination of family name and parent item ID is enforced to be unique. Variant feature names must be unique in the context of their family.

You can create families with the following feature data types:

String (Text) Values in a string type family are always compared lexicographically.

- A text value of 9 does *not* compare equal to "9E0", while a value of 9 compares equal to "9 "(notice the trailing blank). The latter is due to a data type limitation in some database platforms, such as Oracle.
- Text values only support operators “=” and “!=”.
- You can create free-form values for the string type families.
  - The value “” is reserved for internal computations for the string free-form families.
  - Min or max values are not allowed.

Integer Values in the integer families are always compared numerically.

Families where 9 is less than 10 and 9E0 is less than 1E1. The scientific notation of “1.0E2” to represent 100 is a valid integer, but the scientific notation “1E-2” to represent 1% or 0.01 is not valid. The expression “Family>9 & Family <= 10” is equivalent to “Family = 10”.

- Values with this data type are integral numbers. Note that 1.0E2 is a valid integer value (scientific notation for 100), but 1E-2 is not (scientific notation for 1%, or 0.01).

- Valid values are from -2147483648 to 2147483645, even on 64 bit systems.

  This is a default range. It is also applied if the fields are left blank. You can assign your own values only if a desired range is different than the default range.

- 2147483646 is reserved to indicate **Stock Out (SO)**.

- 2147483647 is reserved to indicate **Open Ended (UP)**.

- You cannot create floating point values for integer families.

- 0 is not an allowed value for the integer free-form families because it is reserved for internal computations.

- Floating point values are only allowed for integer families in min or max value fields because a range information may come from engineering knowledge; however, the integer data type may come from production knowledge.

Floating point

Values in the floating point families are always compared numerically.

Families where 9.0 is less than 10.0 and 9.0E0 is less than 1.0E1.

- Values with this data type have 14 significant digits.

- Valid values are 1.7E +/- 308.

Boolean (logical)

Logical value that can have one of two states, for example, On or Off, TRUE or FALSE, and present or absent. Logical values only support operators “=” and “!=”. For example, the TRUE state is represented as "NAME = NAME". The FALSE state is represented as "NAME != NAME".

An empty value assignment such as “MyFlag = "" is equivalent to “MyFlag != MyFlag”, and is therefore a valid assignment even if the family is mandatory.

You cannot add or allocate values to a logical variant feature. Therefore, logical families support neither free-form nor multiselect features.

Values of this data type cannot be compared because families with this data type can only have one single feature such that feature ID and family ID must match.

Date

Families using the ISO 8601 format with time zone identification, for example, families where 2013-09-10 is less than 2013-10-09. In some locales, this may be displayed such that 10.2 is less than 2.10.

- Date values have to be specified and are displayed as Gregorian calendar dates.

- Dates are formatted using a **proleptic** Gregorian calendar.

- Time values are formatted using a proleptic time zone database. That means a time on a date in the past or future is formatted according to the current rules for daylight saving time.
- Date and time are entered in the local time zone.
- Display strings are formatted on the server in server timezone.
- Start dates specify the beginning of the given timestamp, for example, 2016-01-01T00:00:00 specifies the beginning of the 1st second on January 1, 2016.
- End dates specify the end of the given timestamp, for example, 2016-12-31T23:59:59 specifies the end of the last second on December 31, 2016.
- 1900-01-02T00:00:00Z is reserved to indicate **Open Start**.
- 9999-12-30T00:00:00Z is reserved to indicate **Stock Out (SO)**.
- 9999-12-26T00:00:00Z is reserved to indicate **Open Ended (UP)**.

Note:

If preference **TC_Fnd0Booleansolve_EffectivityDateRangeFromTo** is set to the following:

- **FALSE**, the display string does not show seconds, minutes, and hours if they are zero.
- **TRUE**, the display string does not show seconds, minutes, and hours if they are 59, 59, and 23, respectively.

You can set the feature data type and specify the list of allowed features independently. For example, you can create a family with a feature data type of floating point, for which the Product Configurator holds a defined fixed list of valid features.

Why create mutually exclusive family features?

You can define whether families have mutually exclusive features or features where multiple selections can be applied at the same time. For example, a family called **Accessories** may contain features that are *not* mutually exclusive. This allows you to select multiple accessories for a variant configuration solve. If the variant features are defined as mutually exclusive, you can select only one feature at a time.

Typically, most families and their features are mutually exclusive. For example, for a family called **Engine**, the features are typically mutually exclusive and only one feature is valid for a configured

product. You can only select one of the possible engines from the engine family. If you select more than one feature from a mutually exclusive family, then the configuration is invalid.

Distinguishing between mandatory and optional families

An optional (discretionary) family contains product configurations where users are not required to select a feature for the family. For example, you may have a family with features for optional equipment.

It is not necessary to specify a feature from the optional family while configuring a product. When you do not specify a feature, you are allowing Teamcenter to choose a feature for an optional family, based on the system rules.

Conversely, when you define mandatory families, all valid and complete product configurations must select at least one feature for each mandatory family. Leaving a mandatory family unset is equivalent to making an *irrespective of* selection. Assigning an empty value to an optional family (except logical families) is equivalent to making a *none of* selection.

For a mandatory family, you must either specify a feature or a rule to identify a feature from this family. If it is not possible to identify a feature for a required family, then the configuration is incomplete.

You can validate variant configuration criteria against constraint rules managed in the Product Configurator. The Product Configurator provides feedback as to which families are not yet set sufficiently.

Defining multiselect families

You can create multiselect families. A multiselect family implies that more than one selection from this family is possible. They comprise configuration criteria that combine more than one feature to select product data with variant conditions. They may combine features with an AND condition and an OR condition. Only string families can be multiselect. For example, you may have a family called **Accessories** and a valid product configuration where multiple accessory features are selected.

Previous Teamcenter versions supported only single-selection families. Configuration criteria that combine more than one feature for the same single-selection family are interpreted in the same way as the Boolean constant FALSE. Therefore, they select only product data with a variant condition that is also equivalent to the Boolean constant FALSE, irrespective of their use of this family. These criteria do not select product data with variant conditions that combine both these features with an OR condition.

The following example shows configuration criteria referencing the multiple selection family **Accessories**:

Configuration criteria	Description	DesignElement DE1 Accessories = A1 & Accessories = A2	DesignElement DE2 Model = M1 & Model = M2	DesignElement DE3 Accessories != A1 | Accessories != A2	DesignElement DE4 Model != M1 & Model != M2
Accessories = A1 & Accessories = A2		Yes	No	No	Yes
Model = M1 & Model = M2	Criteria are equivalent to FALSE	No	No	No	No
Accessories != A1 | Accessories != A2		No	No	Yes	Yes
Model != M1 & Model != M2	Criteria are equivalent to TRUE	Yes	No	Yes	Yes

In this example, **Model** is not a multiple selection and the default solve type of 529 is used.

> Caution:
>
> You may experience slow responses if you create multiselect families with a large number of members.

Defining dynamic families and standalone features

Your administrator can configure the **Cfg0EnableDynamicFamilySupport** preference to enable creation of dynamic families and standalone features.

> Note:
>
> The standalone features and the dynamic family concept are in an incubation phase. They should only be used with the approval from Siemens product management. They require authoring in Active Workspace and consumption of configurator data.

A dynamic family combines a set of standalone features. The dynamic nature allows you to effectively react to changes in your business process. It provides a mechanism to manage evolution of offered features as it allows standalone features to shift from one dynamic family to another over time. A standalone feature combines the responsibilities of a Boolean feature and its family into a single

business object. You can use standalone features instead of a Boolean family and feature. The standalone feature and dynamic family must belong to the same namespace.

For example, the **Black** and **Red** standalone features belong to both **Exterior** and **Interior** dynamic families.

Car Colors		Group
Exterior	Teamcenter	Dynamic Family
Black	Teamcenter	Standalone Feature
Red	Teamcenter	Standalone Feature
Interior	Teamcenter	Dynamic Family
Black	Teamcenter	Standalone Feature
Red	Teamcenter	Standalone Feature

You create dynamic families similar to a regular literal family. However, you can only add standalone features to the dynamic family.

A dynamic family can be mandatory (one and exactly one standalone feature must be selected) or discretionary (zero or one standalone feature must be selected). You can create an exception rule that allows an exception for a feature under a particular dynamic family by defining an exception condition under which a specified standalone feature does not have mutually exclusive relationship with other standalone features.

Literal feature families group features into a set from which no more than one feature can be selected. With the literal feature families, if you need to efficiently achieve a mutually exclusive behavior, you keep values in a single-select feature family. However, literal features cannot move between families or be taken out of a family to become a standalone feature. Standalone features are arranged into dynamic families with the intent to establish mutually exclusive behavior.

The relationship between a standalone feature and its dynamic family is similar to the relationship between a feature and a feature family. The difference is that standalone feature may be a member of one or more dynamic families concurrently. The relationship between a standalone feature and its dynamic family is managed in a dictionary or a configurator context.

The same standalone feature cannot be a part of multiple groups. It is always optional and can never be free-form or multiselect. Standalone features can be removed from the source and pasted into target dynamic families The same standalone feature can be a part of multiple dynamic families.

You can copy, cut, or drag a standalone feature from the source dynamic family, and paste it or drop it onto the target dynamic family. When standalone features are moved across dynamic families, the system displays a message where you are given a choice to either keep the copy in the source dynamic family or remove the standalone feature from the source. You can select one or more features from the same dynamic family before copying and pasting and dragging and dropping.

If needed, you can release and revise a dynamic family. For example, after revising, you add a new standalone feature to the new revision of the dynamic family. If the previous revision of that dynamic family is configured by the system, then the newly added standalone feature is shown in the **Unassigned Family** group.

> Note:
>
> The **Impact Matrix** view does not support dynamic families and standalone features.
>
> The standalone features and the dynamic family concept are in an incubation phase. They should only be used with the approval from Siemens product management. They require authoring in Active Workspace and consumption of Configurator data.

About numeric features

When you create a numeric feature, under a family type such as an integer or a float, the feature ID and the feature name must be the same. When you provide only the feature ID, the system automatically populates the feature name with the same ID value and vice versa.

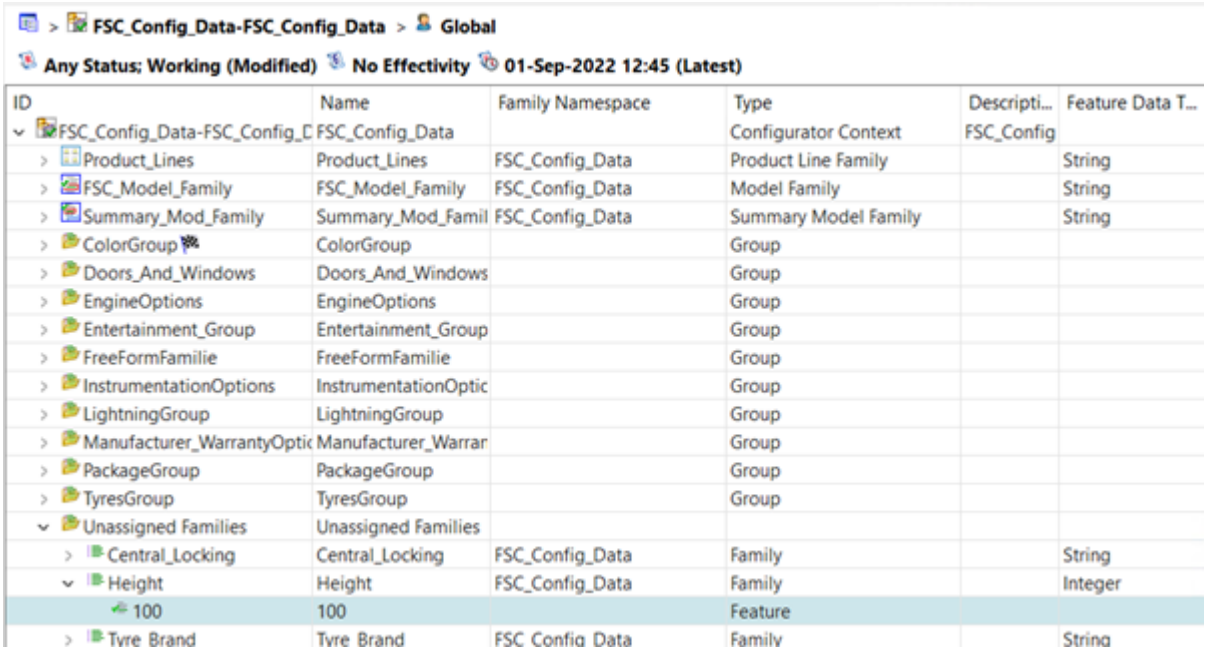
- The system displays an error if the ID and name are different.
- If the ID field is autogenerated using naming rules, from 14.2, the naming rule is not triggered for the numeric feature. The system copies the name you specify to the ID field.
- If the ID field is empty and you specify a name, the name is automatically copied to the ID field.
- If the name field is edited for existing IDs and names prior to 14.2, the system saves the numeric feature only if the name and ID matches.

Define a numeric feature

1. Open the configurator context or dictionary.
2. Select a family of type integer or float under which you want to create the numeric feature.
3. Ensure that the **Name** column is available.

   To add columns, click the **View** menu, select **Column**, and add the required column.
4. Click **Add Feature** and specify a name. For example, if the ID is **100**, the name you specify must be **100**.

   The system displays an error if the ID and name are different.



ID	Name	Family Namespace	Type	Descripti...	Feature Data T...
FSC_Config_Data-FSC_Config_D	FSC_Config_Data		Configurator Context	FSC_Config	
Product_Lines	Product_Lines	FSC_Config_Data	Product Line Family		String
FSC_Model_Family	FSC_Model_Family	FSC_Config_Data	Model Family		String
Summary_Mod_Family	Summary_Mod_Famil	FSC_Config_Data	Summary Model Family		String
ColorGroup	ColorGroup		Group		
Doors_And_Windows	Doors_And_Windows		Group		
EngineOptions	EngineOptions		Group		
Entertainment_Group	Entertainment_Group		Group		
FreeFormFamilie	FreeFormFamilie		Group		
InstrumentationOptions	InstrumentationOptic		Group		
LightningGroup	LightningGroup		Group		
Manufacturer_WarrantyOptic	Manufacturer_Warrar		Group		
PackageGroup	PackageGroup		Group		
TyresGroup	TyresGroup		Group		
Unassigned Families	Unassigned Families				
Central_Locking	Central_Locking	FSC_Config_Data	Family		String
Height	Height	FSC_Config_Data	Family		Integer
100	100		Feature		
Tyre_Brand	Tyre_Brand	FSC_Config_Data	Family		String

5. To create the numeric feature, click **Save**.

Working with auxiliary features

About auxiliary features

Auxiliary features represent simple or complex expressions in a single feature to simplify constraint maintenance. Its definition is created by authoring a separate constraint with the equivalence relation. All other constraints that have used an auxiliary feature in their condition or subject treat it as a substitute of the definition expression. Auxiliary features derive their value from other constraints and do not depend on direct user selections, that is, they do not represent an additional independent point of variation. In that sense, they serve as a lemma. For more information, see **Lemma (mathematics)**.

Auxiliary features are non-business relevant variant points that do not have their own characteristics. They act as a substitute feature in the system. They do not appear in the **Variant Configuration** view when you perform a solve. However, their effects on other regular features are visible during a solve.

The benefits of using auxiliary features are as follows:

- Consumed in constraints as a substitute of its definition.
- The definition can be updated by users at different points in time. Consuming constraints does not need to be modified for immediate effect of updated definition.
- Provides the ability to manage expression which is common across multiple constraints. Changes of common expression are done on an equivalence construct (constraint) that defines the auxiliary feature, and its effect is applied to all the referred instances. This benefits modification at only one place and consumption in multiple places.

- The visibility of the auxiliary feature can be configured in various configurator view such as Constraints Editor and Variant Configuration using preferences (refer details in later sections).

- Can be defined and consumed independently. Users can consume auxiliary feature independently even when its definition is not provided yet. Another user at a later point in time can define auxiliary features using constraints having equivalence behavior (see sections that follow).

> Note:
>
> Currently, auxiliary features are not exposed in variant conditions authoring.
>
> Being non-business relevant features, auxiliary features are restricted for usage in memberships like model, package, and summary. However, there are no restrictions on release, revise, or effectivity operations.

The auxiliary feature has some exceptions to solve behavior and they are as follows:

- Exempted from availability check as they are helper features that do not have any business relevance and only act as substitute expression.

  Therefore, in a negative-biased system, explicit availability is not required to be created for the auxiliary features.

- Does not participate in variant completeness as they do not have any business relevance. The order is complete even if any auxiliary feature is not selected and rest of the features are selected.

Author and manage auxiliary features

Any feature can be qualified as a auxiliary feature based on the **cfg0IsAuxiliary** logical property.

You can create an auxiliary feature in one of the following ways:

- *Using Literal Option Family*: Enable the **cfg0IsAuxiliary** Boolean property on the literal option family to author auxiliary Literal Option features. This makes all its constituent Literal Option values auxiliary.

- *Using Standalone Features*: Enable the **cfg0IsAuxiliary** Boolean property on standalone feature to author auxiliary standalone features.

- *Using Summary Option Family*: Enable the **cfg0IsAuxiliary** Boolean property on the summary option family to author auxiliary summary option features. This makes all its constituent summary option values auxiliary.

> Note:
>
> The default value for **cfg0IsAuxiliary** property is **false** for all the types mentioned above.

This property is not visible on any other family type like package option family or model family Hence, auxiliary behavior is not applied on such types.

Modify the auxiliary property

The **cfg0IsAuxiliary** Boolean property is modifiable.

- Allowing changes to auxiliary property helps in reusing existing features as auxiliary features.
- No restrictions to modifying the **cfg0IsAuxiliary** property for literal and summary features. However, for standalone auxiliary features, changes are restricted if the change causes an inconsistent auxiliary state among all standalone features within a dynamic family.

The system validation fails in the following cases when you modify the **cfg0IsAuxiliary** property:

- On a standalone feature to **false** whose peer standalone features have **true** value for the property.
- On a standalone feature under a dynamic family that is already a part of a different dynamic family.

Note:

Updating the **cfg0IsAuxiliary** property of all standalone features under a dynamic family to the same value in one go is allowed only in the Active Workspace client.

Moving standalone auxiliary features across dynamic families trigger validations. If you move an auxiliary standalone feature to a dynamic family where all the existing standalone features are non-auxiliary, then the system restricts this operation. It displays an error as this change can lead to an inconsistent auxiliary state.

Visibility control of auxiliary features

The visibility of the default auxiliary features in different Active Workspace views is as follows:

View name	Default visibility	Configurable?
Variability Explorer	Yes	No
Constraints	Yes	Yes, using **Cfg0ShowAuxiliary.CONSTRAINTS** preference.
Variant Configuration	No	Yes, using **Cfg0ShowAuxiliary.VARIANT_CONFIGURATION** preference.
Variant Conditions Authoring	No	No

- **Cfg0ShowAuxiliary.CONSTRAINTS** is a site level preference that controls the visibility of the auxiliary feature in the **Constraints** view. The default value of this preference is **true**. End users can override this as needed.

- **Cfg0ShowAuxiliary. VARIANT_CONFIGURATION** is a site level preference that controls the visibility of the auxiliary feature in the **Variant Configuration** view. The default value of this preference is **false**. End users can override this as needed.

Define auxiliary feature

The auxiliary feature's definition can be defined using constraints. It must have equivalence behavior.

There are two ways to define auxiliary features:

- Variability

Family	Feature	Is Auxiliary
Model	**LXI, VXI**	Not applicable
Sunroof Type	**Normal Panoramic**	Not applicable
Wheel Type	**Alloy, Steel**	Not applicable
Model Type	**Standard, Intermediate, Advanced**	**Y**

  Till this point, auxiliary features (**Standard**, **Intermediate**, **Advanced**) are authored. However, they do not have any definition.

- Availability constraint

  Create availability constraint to make Auxiliary feature available to regular features.

  The system imposes implicit equivalence relationship for the auxiliary feature when used as subject in the Availability constraint.

  Examples:

Availability Constraint	Condition	Subject
Rule1	**Normal AND Steel**	**Standard**

  With these rules, the definition is provided for the Standard feature.

  Definition: **Standard == Steel AND Slide**

If the user has multiple Availability constraints with the same auxiliary feature, then the system implicitly converges and creates one definition using all availability constraints of that auxiliary feature.

Availability Constraint	Condition	Subject
Rule1	**Normal AND Steel**	**Standard**
Rule2	**Alloy**	**Standard**

Definition: **Standard == (Normal AND Steel) OR Alloy**

The benefit here is that multiple users can define their own business intent of auxiliary features separately. The auxiliary feature's consumer constraints bring in effect of definitions from all availability constraints.

- You can define the auxiliary feature using the custom type Include constraint having equivalence behavior.

  > Note:
  >
  > You must make sure that there exists only one definition for an auxiliary feature.

- The availability constraint must not be created for auxiliary features in a negative biased context. This is because the Availability constraint is going to be considered as definition of auxiliary features.

Solve behavior for the auxiliary feature in the configuration

The auxiliary feature is not visible in the **Variant Configuration** view. It is exempted from the availability check. Moreover, constraints created using the auxiliary feature have an impact when users perform validation or expansion.

Example:

Create a negative-biased context with the following variability:

Family	Features	Is Auxiliary
Model	**LXI, VXI**	Not applicable
Sunroof Type	**Normal Panoramic**	Not applicable
Wheel Type	**Alloy, Steel, Diamond**	Not applicable
Model Type	**Standard, Intermediate, Advanced**	**Y**

Constraint	Type	Condition	Subject
Rule1	**Availability**	**Steel**	**Standard**
Rule2	**Availability**	**Alloy**	**Standard**
Rule3	**Availability**	**Panoramic AND Diamond**	**Advanced**
Rule4	**Include**	**VXI**	**Advanced**
Rule5	**Default**	**LXI AND Standard**	**Normal**

Use case 1

User selection: **VXI**

Output: **Valid & Incomplete**

Expanded expression: **VXI AND Panoramic AND Diamond**

Constraint trace for expand result:

1. User selection: **VXI**
2. Rule4 (Include Constraint) – **VXI includes Advanced** (Auxiliary Feature)
3. Rule3 (Availability Constraint – 'Advanced' Auxiliary feature's definition) – **'Advanced' is equal to 'Panoramic AND Diamond'**

Use Case 2

User selection: **LXI AND Alloy**

Output: **Valid And Complete**

Expanded expression: **LXI AND Alloy AND Normal**

Constraint trace for expand result:

1. User selection: **LXI AND Alloy**
2. Rule1/Rule2 (Availability Constraint – 'Standard' Auxiliary feature's definition) – **'Standard' is equal to 'Steel OR Alloy'**
3. Rule5 (Default Constraint) – **'LXI AND Standard' defaults 'Normal'**

Use case 3

User selection: **LXI AND Diamond**

Output: **Valid And Incomplete**

Expanded expression: **LXI AND Diamond**

Solve example using auxiliary features

Create a negative-biased context with the following variability:

• Models:
  • **LXI**
  • **VXI**
• **Infotainment** [single select family]
  • **Touchpad**
  • **Stereo**
• Feature Collection [Multiselect]
  • **Aux cable**
  • **USB**
  • **BT**
• Auxiliary [multiselect]
  • **X_info**
  • **X_Nav**
  • **X_None**
• Wheel Type
  • **Steel**
  • **Alloy**
  • **Diamond**
• **X_Fam**
  • **X_Steel**
	Availability:
	Consider that you have created availability constraints for all non-auxiliary features for **LXI** as well as **VXI** model. No Availability is required for the Auxiliary feature as it is excluded from the availability check.
	• **X_steel -> alloy [x_steel available for alloy]**
	• **X_steel -> diamond [x_steel available for diamond]**
	The system creates the following simplified relation:
	X_steel <-> alloy | diamond
	Negative version:
	!X_steel <-> !alloy & !diamond
	Include Constraint:
	• **IR1: Stereo-> X_info**
	• **IR2: Steel -> Stereo**
	• **IR3: X_Info-> USB & Aux**
	Default Constraint:
	• **DR1: LXI-> stereo**
	• **DR2: VXI & !X_Steel-> Steel**

Use case for LXI selection

- User selection: **LXI**

- Output: **Valid & Incomplete**

  Expanded expression: **LXI & Stereo & USB & aux**

Due to the default constraints, **DR1stereo** is selected and due to stereo **IR1** the system brings **X_info**. As it is an auxiliary feature, it is not seen in the configuration view; however, its definition provided by **IR3** is exercised and **USB & Aux** is selected.

Use case for VXI selection

- User selection: **VXI**
- Output: **Valid & Complete**

  Expanded expression: **VXI & Steel & Stereo & USB & aux**

Due to the back propagation of availability constraint, **! X_steel** is selected, and it triggers **DR2** and brings the selection for **Steel**. This in turn exercises **IR2** and brings in **Stereo**. Then it triggers **IR1** and brings **x_info** auxiliary feature. As it is an auxiliary feature, it is not seen in the configuration view; however, its definition given by **IR3** is exercised and **USB & Aux** is selected. As there are selections available for all families, the expanded expression is complete.

Use case for LXI and Alloy selection

- User selection: **LXI & Alloy**
- Output: **Valid & Complete**

  Expanded expression: **LXI & Stereo & USB & aux**

Due to the default constraints, **DR1stereo** is selected and due to stereo **IR1** the system brings **X_info**. As it is an auxiliary feature, it is not seen in the configuration view. However, its definition given by **IR3** is exercised and **USB & Aux** is selected. As there are selections available for all families, the expanded expression is complete. Here, though there is no selection for **X_Fam** auxiliary feature, the system provides a complete verdict since the auxiliary feature does not participate in completeness.

Grouping families

You can group families to make it easier for the user to create variant configuration criteria. Presenting families for which you must assign values in manageable sets reduces the complexity of working with the variant model. This approach also reduces the complexity of the techniques that can be employed to compute available variability. The effort to filter the list of available features based on all constraints grows exponentially with the number of families and the number of constraints. Response times may be acceptable for manageable sets of families, but may lead to unacceptable response times if you have a large numbers of families or constraints.

You can use Product Configurator to browse and edit the list of groups and their families.

After allocating variability for a family to a configurator context, you can group these families. It is not compulsory to assign any family to a particular group. You manage the grouping at the configurator context level; the same family may be assigned to different groups in different configurator contexts.

> Note:
>
> Within a configurator context, a family may exist in only one family group.

You can release and revise a family. But you cannot revise a group. The system always displays groups because they are excluded from the revision-based search.

A group is not a configuration construct like a family or a feature. For example, if the revision rule is set to **Any Status; No working**, the system displays all available groups in the **Variability Explorer** irrespective of their status, both in a working or a released state.

Configuration management of variant data

As variant data is introduced and changed, Teamcenter ensures the correct versions of the data are considered and offered when you select family features and configure the product data.

It is important that your business processes ensure the appropriate product data, families, and variant rules are provided in a coordinated way. Teamcenter helps you understand what changes you have already accounted for in the variable product definition and what changes have yet to be considered. That is, you should understand if you have introduced new features that are not referenced elsewhere and also if features they are obsoleting are referenced in active rules.

Careful configuration management is important if you are performing any of the following tasks:

- Introducing families, features, and applicability

  As marketing specialists or product managers introduce new families and features, you may need to define business rules that determine what level of maturity or approval is needed before these families and features can be accessed by users.

  - Validated for applicability for a product model (that is, a model-feature mapping can be created for them).
  - Eligible to have rules written to control their selection.
  - Referenced directly by product data.
  - Exposed to a user for configuration.

  Typically, you enforce these business rules with access rules on the relevant objects to allow only authorized users to view and manage the data.

You may define additional business rules to determine eligibility to have existing rules exercised during a configuration process. For example, when a user first creates and tests variant rules, they do not impact the existing variant rules and product data until they are validated, mature, and effective.

In addition to defining and enforcing access rules on the variant expressions, you should control when new families, features, model-feature applicability, and variant rules created by a user should become available, applicable, or enforced. One method of doing this is with a release process.

Alternatively, you can schedule the availability of families and features or the applicability of rules with effectivity. This allows you to differentiate between the effectivity of variant data for authoring or testing and the availability of the same data to a user.

For example, if you introduce a new color paint for one of the product models, you would make it available for testing and validation before making the new color feature a selection that is visible to the user.

- Evolving families, features and applicability

  As families and features evolve (that is, metadata or attachments are updated), Teamcenter carries forward any product model applicability or references to these families in variant rules or via references from product data.

  Because it may not be enough for a new feature to become effective at a particular time, you can control its introduction to each product model for which it is authorized.

  As variant expressions or rules evolve, it allows you to understand the changes to validate the update is accurate and complete. Likewise, you can control when any changes to variant rules become effective.

- Obsoleting families, features, and applicability

  When you obsolete families, features, and applicability, you can also control when the families and features cease to become available, and also when the rules cease to be available.

- Creating and managing variable product data

  The creation and updating of the product data and the variant rules and families that are referenced directly by the product data are straightforward to manage. Typically, your company already has an established business process for the management of product data that requires the revising of the product data when changes significant to the business occur. The same process can normally be used for variable product data.

  You manage changes that are significant to the business (for example, adding or removing referenced families or updating variant rules) by making these changes only to an editable version of the product data. A specific revision of the product data references a feature and (as that same feature evolves), the latest version that meets the defined business rules applies. This allows the user to make form, fit, and function compatibility decisions when making changes to a feature. If its interface definition does

not change, the product data and rules that reference it should not change. If the interface definition changes, you should introduce a new feature or revise the product data that references it.

- Reorganizing your products and features to supporting evolving product suite

  Over time, product portfolios evolve, companies are acquired and merged, and the amount of product variety continually increases. In addition, the amount of variability for a given area of the product may grow and can be combined in different ways. You can standardize and reuse features and rules or rearrange the product portfolio to merge previously separate products.

  For example, in the early days of managing a heavy equipment product line, there are only a small number of available engine types, so these are all grouped together. Over time, as the number of engine types grows, it may be necessary to organize these into more specialized groups. The system recognizes the equivalence of a given feature regardless of how it is organized.

Defining families, features, and configurator rules

When you specify the variability to offer on products, you work with the following objects:

- Families

  Users perceive these as the questions asked during the configuration process.

- Features

  Users perceive these as the possible answers to the questions asked during the configuration process.

  Each feature refers to a family it is associated with. The **Family** property for each feature displays the related family. That property cannot be modified. A family does not contain the information of how many and which features are related to it.

- Saved variant rules or variant constraints (inclusion rules, exclusion rules, default rules, and availability rules)

  Users perceive these as expressions of the valid or invalid combinations of allowed answers.

When defining variant data, you must be able to:

- Identify families that are offered for your product line. Family or feature choices can be developed by deciding the question that the user is asked at each step of the process. For example:

  - Color: What color car would you like?

  - Sunroof: Would you like a sunroof?

- Define allowable features for each family. Features are all the possible answers to the questions the user is asked. For example:

  - Color: cayenne red, dusty blue, charcoal grey

  - Sunroof: yes, no

In the previous examples, there is one expected answer for each question. That is, these questions are mandatory for a complete, valid configuration and the answers are also mutually exclusive. Consequently, only one answer may be given.

There are also situations where no answer or multiple answers to the same question are valid for a certain configuration. For example:

- CD changer: 1 disc, 6 disc, 10 disc.

  This is an optional (discretionary) family. The user need not make any selection. If the user does not specify a feature during configuration, Teamcenter chooses a feature for an optional family, based on the system rules. If the user does make a selection, only one selection is allowed. Consequently, these features are mutually exclusive as well as discretionary.

- Cold weather accessories: snow chains, engine block heater, heated seats, remote starter.

  This is also an optional (discretionary) family, but in this case the user may select more than one feature. These features are not mutually exclusive.

You can make all of the families mandatory and all features mutually exclusive using a configurator rules. For example:

```
CD changer: 1 disc, 6 disc, 10 disc, none
Snow chains: yes, no
Engine block heater: yes, no
Heated seats: Driver only, driver and passenger, none
Remote starter: yes, no
```

You can also make the different families themselves mutually exclusive. For example, if the user selects cold weather accessories, they cannot select warm climate accessories. Likewise, if the user selects split rear seats (60/40, 40/20/40), they cannot select a rear cargo barrier, aluminum lattice, elastic mesh, or vertical bars.

You want to establish other types of rules or constraints among the different families or features. Also, you may want to create more sophisticated logic between the different families or features. For example:

- Rules defining what selections may not be made in combination. For example, if you select **Body Color = Green**, you cannot select **Interior Color = Blue**. For example:

```
Report message "Your Message"
with severity "Your Severity" when violated.
"[CAR]Bluetooth = true" is not allowed when "[CAR]Drive = AWD".
```

- Default features. You can define rules that select a default feature as a starting point or the commonly expected choice. Default rules may be defined independently of the features chosen for any other family. Unless overridden by the user or by other constraint rules with a higher severity, the system evaluates the default rules to select the appropriate feature.

  For more information, see **About default rules** and **Understanding severity levels in configurator rules**.

- Derived features. In some situations, the user may want to define a rule where the setting or default feature of one or more features is a direct result of other feature selections. These might be fixed or overrideable. For example, if you select sport suspension, you must also select the performance tires. This could be expressed by exclusion rules or error checks, for example:

  ```
  (‘(if Suspension = Sport ErrorIf Tires = All Season)
  OR (if Suspension = Sport ErrorIf Tires = RunFlat)
  OR (if Suspension = Sport ErrorIf Tires = SnowTires)’)
  ```

  However, it is more efficient to express this dependency as
  `The default configuration is "Tires=Performance" when "Suspension=Sport".`

  You can express the same type of dependency but allow the user to override it. For example:

  ```
  The default configuration is "HeatedSteeringWheel = Yes"
  when "(Heated Seats = Driver | Heated Seats = Driver and Passenger)".
  ```

  In this example, the heated steering wheel is a default or suggestion that is presented to the user only if they have made other complementary choices. The user may still decline the heated steering wheel and to change the default to `No`.

Features can move over time to different families. That is, the family with which a feature is associated is not static and can evolve.

You can explicitly validate a feature, on demand, to check for data consistency by clicking **Validate Variant Options** in the **Variability Explorer** view. You can also select a feature and choose **Validate Variant Options** from the shortcut menu. If the validation fails for a feature, that feature is shown in red and the validation message appears. The failed feature lists the tooltip with the violation details.

Writing variant expressions and solve criteria

If product data does not have a variant condition, some businesses consider an error has occurred. Even if a component is always configured, your business process may require an explicit statement to always configure this component. In this scenario, it is important to find objects that do not have a variant

condition, otherwise you cannot ascertain if an object configures for all variant criteria because it was forgotten or because it is intended always to configure.

The following table illustrates possible variant configuration scenarios, the configurable objects and their variant condition. Each row represents a variant solve configuration using the criteria shown in the criteria column. Columns A through E represent objects configured with the variant conditions shown in the condition row. For example, a cross (X) in cell A,1 indicates that configurable object A configures for the criteria in row 1.

Criteria →		**A**	**B**	**C**	**D**	**E**
Condition →		**ENG !=V8**	**ENG=V8**	**ENG=V6 OR ENG! =V6 (TRUE)**	**ENG=V6 AND ENG ! =V6 (FALSE)**	**None**
1	Formula “ENG=V6”	X		X		X
2	Formula “ENG! =V6 OR ENG! =V8”	X	X	X		X
3	Formula “ENG=V6 AND ENG=V8”					X
4	State “TRUE”			X		
5	State “FALSE”				X	
6	State “NULL”					X
7	State “NOT_NULL”	X	X	X	X	
8	State “NON-CONTSNT CONDITION”	X	X			

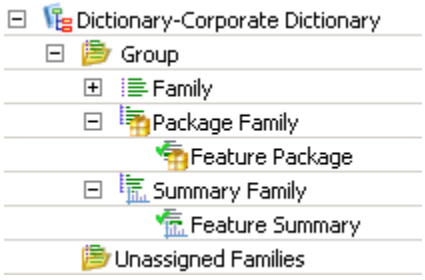
The following table shows the configuration conditions in the previous example, with the corresponding solve types.

Configuration condition (cell reference)	Solve type
A1; B1	Configure objects with a variant condition against variant solve criteria.
C1	Configure objects with a variant condition that is equivalent to the Boolean constant TRUE against variant solve criteria.
D1	Configure objects with a variant condition that is equivalent to the Boolean constant FALSE against variant solve criteria.
E1	Configure objects without variant conditions against variant solve criteria.
A2; B2	Configure objects with a variant condition against variant solve criteria that are equivalent to the Boolean constant TRUE.
A3; B3	Configure objects with a variant condition against variant solve criteria that are equivalent to the Boolean constant FALSE.
C2	Configure objects with a variant condition that is equivalent to the Boolean constant TRUE against variant solve criteria that are equivalent to the Boolean constant TRUE.
D2	Configure objects with a variant condition that is equivalent to the Boolean constant FALSE against variant solve criteria that are equivalent to the Boolean constant TRUE.
C3	Configure objects with a variant condition that is equivalent to the Boolean constant TRUE against variant solve criteria that are equivalent to the Boolean constant FALSE.
D3	Configure objects with a variant condition that is equivalent to the Boolean constant FALSE against variant solve criteria that are equivalent to the Boolean constant FALSE.
E2	Configure objects with no variant condition against variant solve criteria which are equivalent to the Boolean constant TRUE.
E3	Configure objects with no variant condition against variant solve criteria that are equivalent to the Boolean constant FALSE.
C4	Configure objects with a variant condition that is equivalent to the Boolean constant TRUE against variant criteria selecting objects with a variant condition that is equivalent to the Boolean constant TRUE.

Configuration condition (cell reference)	Solve type
D5	Configure objects with a variant condition that is equivalent to the Boolean constant FALSE against variant criteria selecting objects with a variant condition that is equivalent to the Boolean constant FALSE.
E6	Configure objects with no variant condition against variant criteria selecting objects without a variant condition.
A7	Configure objects with a variant condition against variant criteria selecting objects with a non-NULL variant condition.
B7	Configure objects with a variant condition against variant criteria selecting objects with a non-NULL variant condition.
C7	Configure objects with a variant condition that is equivalent to the Boolean constant TRUE against variant criteria selecting objects with a non-NULL variant condition.
D7	Configure objects with a variant condition that is equivalent to the Boolean constant FALSE against variant criteria selecting objects with a non-NULL variant condition.
B8	Configure objects with a variant condition against variant criteria selecting objects with a non-constant variant condition.
A8	Configure objects with a variant condition against variant criteria selecting objects with a non-constant variant condition.

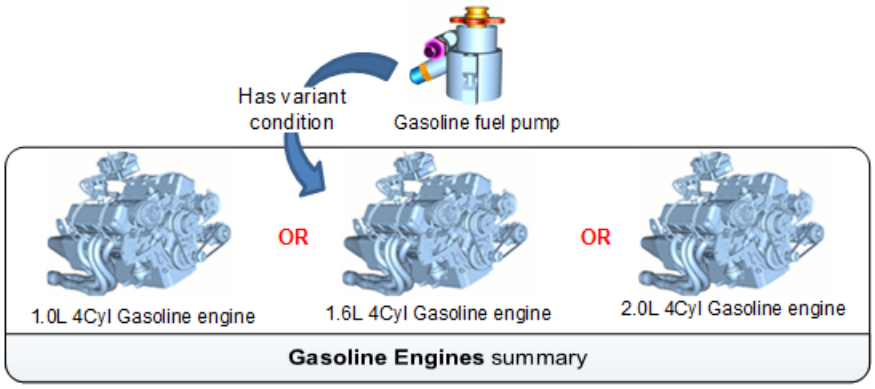
Defining summaries

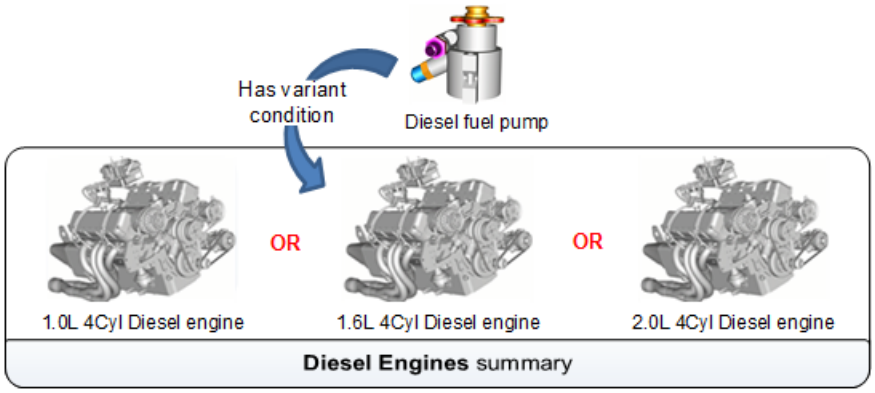
Summary family is a collection of feature summaries which allows you to summarize the features from the same family. Feature summaries are connected using a logical OR operation in the variant expression. Features may belong to more than one feature summary.



For example, a generic car engine uses the same gasoline fuel pump for all gasoline model engines and the same diesel fuel pump for all diesel engines. The configuration analyst, using Product Configurator,

creates the **Gasoline engine** feature summary that contains all gasoline engine models and the **Diesel engine** feature summary that contains all diesel engine models.



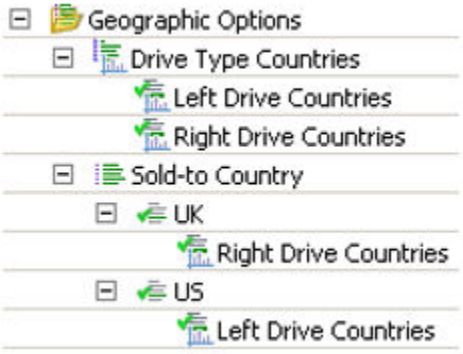


Feature summary allows you to efficiently author and maintain configurator rules by referring to the *feature summary* when the rule applies to all of the features it summarizes. A feature summary may be referenced in variant, exclusion, and inclusion, and availability rule expressions.

You can only summarize features within the same mutually exclusive family. You cannot summarize features from the following:

- Across mutually exclusive families
- In a multiselect family
- From a Boolean family
- From a free-form family

Feature summaries are typically used to represent sets of alternative choices. For example, **Left Drive Countries** feature summary may summarize US, Canada, and Germany.

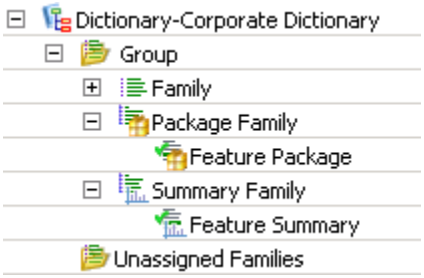


Feature summaries cannot be ordered by a customer, thus they are only available when configuring with a custom variant configuration using *manual validation* and *not* available when configuring using *guided validation*. After selecting a feature summary, all the features that are valid members are selected when you validate the configuration using manual validation.

You can copy a feature summary using the **Save As** command. All summarized features from the source feature are carried forward to the new feature summary.

Group features to create packages

You can group features to create *feature packages* that you would like customers to order together in a bundle. *Feature packages* contain features within or across families. Existing features are assigned as members of a package and are connected using a logical **AND** operation in the variant expression.



Feature packages are typically used to represent sales or marketing packages, such as a luxury or economy package for a car. Similar to feature summaries, feature packages allow you to efficiently author and maintain configurator rules when those rules apply to all of the features within the package.

Typically, a product manager or similar user identifies one or more of the following reasons to group families in a bundle.

For example, the following **Cold Weather Package** contains the selected **Engine Block Heater** and snow tires.

Special Packages	Group
Cold Weather Package	Package Family
Falken Ziex ZE-Ice	Feature Package
215/50/17 Falken Ziex ZE	Feature
215/50/17 Falken Ziex ZE-Ice	Feature
Engine Block Heater	Feature
Kumho IceSlayer	Feature Package
215/50/17 Barum Bravuris 2	Feature
215/50/17 Kumho IceSlayer	Feature
Engine Block Heater	Feature

- You want to offer a wide range of variability to your customers, but recognize there are efficiencies and cost savings in more commonality.

- You recognize that customers want to be able to order and receive their products quickly. Defining set configurations allows you to produce common product variants as needed and keep stock on hand. Offering feature packages instead of complete configured variants allows you to maintain a balance between permitting the customer to select features they want dynamically without forcing them to choose only from fully specified configurations.

- There are features that are easier or harder to accommodate late in the process. For example, a laptop computer is available in different colors, but installation of the colored cover is the last step before it is boxed and shipped to the purchaser. You may have several configurations built and stocked that are only waiting for the cover color matching incoming orders to facilitate the final assembly of those products.

- You want to encourage the customer to order more accessories or premium features. By grouping them in an easy to order package, you may achieve this objective. You may also offer a discount over ordering each feature individually.

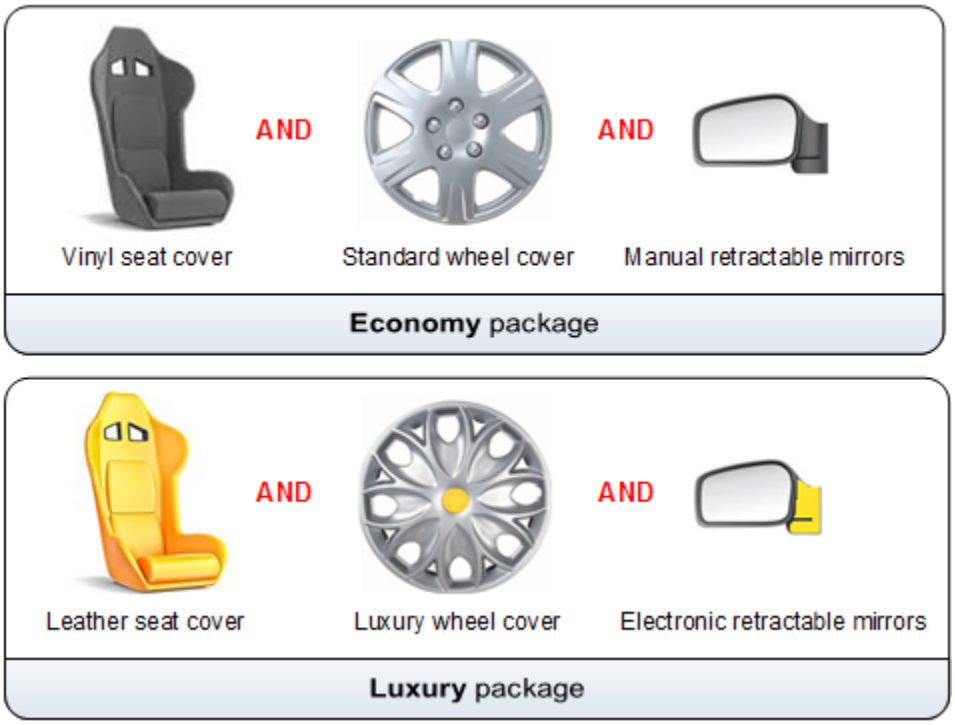
- By creating repeatable configurations or partial configurations, you reduce the likelihood of issues with a particular combination of features.

If you create packages to bundle sets of features, you should also optimize your other business processes around the use of feature packages. That is, considerations such as engineering feasibility and the creation of engineering solutions should be partially determined from the context of the feature package, rather than individual features separately.

Feature packages are exposed during the configuration process. If you select a package, Teamcenter responds by selecting all of the corresponding features collected in that feature package when you validate the configuration using manual validation or guided validation. However, selection of package members does not result in selection of packages.

For example, the economy feature package for a generic car includes a vinyl seat cover, standard wheel cover and manual retractable mirrors. The luxury feature package for a generic car includes a leather seat cover, luxury wheel cover and electronic retractable mirrors. The configuration analyst,

using Product Configurator, creates the luxury and economy package features that contains the features for each package.



Distinguishing between marketing and technical features

There are two main categories of features to consider when defining and managing product variability:

- Marketing features. These features are expressed in terms easily understood by the end user or customer. They describe the feature or benefit obtained by selecting that feature.

- Technical or manufacturing features. Those features describe some characteristic or variability that is required for technical or manufacturing reasons. They are normally not directly relevant to, exposed to, or understood by the customer.

A marketing representative or product manager typically determines the product variability to offer to the marketplace. Often, but not always, this decision is presented in terms of the specific product you are developing or planning. The terms used to describe these features are ones that resonate with a sales engineer or customer, and describe what capability or benefit is obtained by choosing that feature. Usually, the process of deciding the features to offer to the marketplace occurs separately from development of the actual parts and product data that will support the physical delivery of those features.

Only marketing features are available to an end user, not any technical features, and not product or BOM data. You can make a complete, valid set of feature selections that result in a valid 100% BOM from only the marketing families and features.

The process of evaluating and deciding the features to offer may include technical or feasibility input from engineering. However, at this stage of product planning, there are often no design solutions designed to support the features suggested.

Once the features to offer and any rules that restrict or mandate particular combinations are defined, engineering typically completes the design work necessary to support the features. That is, technical personnel identify, certify, or develop the design solutions that become the physical parts delivering the specified feature.

There may be technical parameters or features implied by the marketing features exposed to the end user. For example, `Operating Market = United States` may have implications for the voltage characteristics of the electrical components in the product. This relationship or dependency is expressed by creating derived rules from the marketing family to the technical families it implies. In this example, the technical families are referenced directly from the product data, and included in the variant conditions.

These marketing and customer facing features and their characteristics may be captured in a data management system, or they may be largely or partially captured in a separate external configurator. For example, basic feature pricing may be captured in a data management system, but complex pricing or discounting, including geographic dependencies, are typically managed in an external sales configurator.

For any marketing or customer feature information captured in Teamcenter, the marketing specialist or product manager has significant additional flexibility to involve product specialists as they consider the technical feasibility of proposed offerings. Both the marketing users and the engineers participate in the review and approval process.

Sales configurators generally lack the configuration management capabilities of Teamcenter. Teamcenter supports controlled evolution, history, and release of new families, features and rules where the sales configurator does not.

In addition, the marketing features may be directly referenced by the product data. That is, the marketing feature is directly referenced by the product data and also directly referenced in a variant condition that controls product data applicability.

Distinguishing between business intents

You can categorize configurator objects with one or more business purposes or intents, such as technical, marketing, and so forth. The intent of a feature can evolve over time. You can define intents on configurator objects, such as features, families, configurator rules, and groups.

An intent is the purpose or reason why an object was authored or used. For example, features can be purely technical, for example, features for air conditioner tonnages as 2 tons, 4 tons, and so forth. Additionally, features can be used for marketing and sales configurator, for example, a limited edition package. Features can also represent an engineering artifact, for example, an engine.

In Product Configurator on Rich Client — Usage, you can identify and work with configurator objects according to their function and need in the system. Some features can have multiple intents and can evolve and change their intents over time. Therefore, the intent is the subject of an object revision, for example, multiple revisions of an object can have different sets of an assigned object intent.

> Note:
>
> An intent of an object does not change the solve behavior. Intents are used only to categorize, filter, and choose the variability set for various validations.

You can add or modify intents only using the **Edit Properties** dialog box.

By default, you can assign the following intents:

- Manufacturing
- Marketing
- Technical

> Note:
>
> Your administrator can add your company-specific intents in the Business Modeler IDE.

Defining applicability of families and rules

You can control the reuse of families and features by designating certain families or subsets of features that are allowed for a particular product or product model.

As you introduce new families, you can reuse them across several product models. However, you can restrict the product model applicability of features to only the desired product models. As a result, if you introduce a new product model, the user can identify the features to offer for that product model.

You can use the Product Configurator to view and manage all product models and features in a single matrix. Alternatively, you can build a table with only the product models and features of active interest.

You can identify product model-feature pairs to authorize. When you author saved variant rules or configure selected features in the context of a product model, only those features authorized for the selected product model are shown.

Defining availability of features

As you introduce new product models, product lines, or summary models, you can define which features, families, and feature packages are available for the product models within a configurator context.

There are two ways you can define availability of the variant data for your product models, product lines, or summary models:

- Create availability rules using the **Configurator Rules** view.
- View and manage all product models, product lines, or summary models and all available features and families in a single **Availability Matrix**.

  You can either:

  - Choose each feature.
  - Drag many configuration features and drop them onto a column header for a model in the matrix.

  You can select the set of models for which features should be available. Teamcenter creates an availability rule for each selection of a feature in the column for a given model. This maximizes flexibility because the availability is managed for each model individually. Using a single **Availability Matrix**, you can author a large set of individual availability rules.

Allocating and deallocating variant data

Allocation

You can allocate each group, family, and feature to one or more configurator items. The objects are shared across configurator items.

- A feature is automatically allocated to the configurator context or dictionary when it is created, together with its corresponding family and group.
- When you allocate a feature to a configurator item:
  - Teamcenter checks if the feature's family is allocated to the target context. If yes, then it allocates the feature.
- When a feature is selected for allocation, only the selected feature is allocated. Other features within the same family are not allocated.
- When you allocate a family to a configurator context:
  - Teamcenter checks if a group to which the family belongs to is allocated to the target. If yes, the family appears under the group, else it appears under the **Unassigned Families** group.
  - The family’s features are not allocated to the target configurator context.
- When you allocate a group to a configurator context:

  - Teamcenter ensures only the group is allocated to a target configurator context item.
  - Families and features are not allocated to the target configurator context item.
- When you allocate a summary family to a configurator context:
  - Teamcenter checks if a group to which the family belongs to is allocated to the target. If yes, the family appears under the group, else it appears under the **Unassigned Families** group.
- When you allocate a feature summary whose summarized feature is not allocated to the target configurator context:
  - The selected feature summary is allocated to the target, if its summary family is allocated to the target configurator context.
  - If new features are added to the source summary family, these new features are not automatically allocated to the target configurator context.
- When you allocate a package family from a source dictionary to a target dictionary or a configurator context, Teamcenter:
  - Allocates a new package family to a target dictionary or a configurator context.
  - Checks if a group to which the family belongs to is allocated to the target. If yes, the family appears under the group, else it appears under the **Unassigned Families** group.
- When you allocate a feature package from a source dictionary to a target configurator context:
  - The selected feature package is allocated to the target if its package family is allocated to the target configurator context.

You can select all families in a group and all features in a family for allocation when you right-click the group or the family, respectively, and choose **Expand and Select**  .

Teamcenter does not support updating of a configurator allocation. For example, if you allocate a group from a source item to a target configurator context and then add a new family to the group in the source, Teamcenter does not automatically update the family allocations. You must manually allocate a new family to the target configurator context.

After allocating variant data to a target configurator context or a dictionary, the system refreshes the contents of the target to display the newly allocated objects.

If the target view contains a rule date, the system sets the most recent time as a new rule date and refreshes the contents, which produces the same result when you manually reload the view by clicking **Reload View**  .

If the target contains **No Rule Date** configuration, the system refreshes the contents without changing the rule date.

Deallocation

- When you deallocate a feature from a dictionary or configurator context:
  - Teamcenter validates that the feature is not referenced in any variant expressions for rules or in any compound features, such as feature packages or feature summaries, associated with the configurator context.
- When you deallocate a feature or a summary family from a dictionary or a configurator context:
  - Teamcenter validates that the feature or summary family has no allocated features in the dictionary or the configurator context.
  - If the family is free-form, validates that no free-form values are referenced in any variant expressions for rules associated with the dictionary or the configurator context.
- When you deallocate a group or a package from a dictionary or a configurator context, Teamcenter:
  - Validates that the group has no allocated families in the dictionary or the configurator context.

> Note:
>
> Before deallocating a dynamic family, ensure the Standalone features under it are deallocated.

Working with allocation

By setting your Product Configurator on Rich Client — Usage **Variability Explorer** view to **Context** mode, you can view and work with configurator allocation. Configurator allocation is a workspace object, and it can have its own life cycle. Configurator allocation can be configured using effectivity. It can also have multiple revisions.

In **Context** mode, you can perform the following actions on configurator allocations, based on a license applicable for an actual object. All other actions that are available in the **Variability Explorer** view are disabled by default in **Context** mode.

> Note:
>
> To view actual objects, such as features, families, and groups, set the **Variability Explorer** view to **Global** mode.

Action	Actual Object Global mode	Configurator allocation Context mode
Set effectivity	Enabled	Enabled
Revise	Enabled	Enabled
Submit to workflow	Enabled	Enabled Any configurator allocation record irrespective of its target type can be submitted to a workflow.
Cut	Enabled In **Global** mode, it removes the configurator allocation of selected object from the configurator context.	Disabled In **Context** mode, it is disabled because the configurator view displays relation objects.
Delete	Disabled In **Global** mode, it is disabled because you cannot delete persistent feature data in the **Variability Explorer** view.	Enabled You can delete a selected configurator allocation record.
View or update properties	Enabled	Enabled Teamcenter internally copies most of the applicable properties from an actual object to the applicable configurator allocation when the allocation is initially established. Common properties between a configurator allocation and actual object are merged with precedence given to the allocation. If a property is empty on a configurator allocation, then the property value from an actual object is used for display. An **ID** column is not merged. Properties that are not available on the configurator allocation are read-only. Only the **Variability Explorer** view displays merged properties. All other views show the properties directly from a configurator allocation.
Copy and paste	Enabled	Enabled You cannot paste in the **Variability Explorer** view when the clipboard object is of the configurator allocation type. Teamcenter generates an error message if a configurator

Action	Actual Object Global mode	Configurator allocation Context mode
		allocation is used for pasting onto the **Variability Explorer** view.
Drag and drop	Configurator allocation is created with respect to the target context	Teamcenter ignores allocation if there are multiple objects selected for paste. Otherwise, Teamcenter generates an error stating that allocation is not supported for the configurator allocation objects.

Deleting variant data

You can delete groups, families, and features only if they do not reference any configurator items and are not referenced by any configurator rules or variant expressions. You must have delete access to the relevant objects.

You cannot delete a family if it contains any features.

You can delete configurator rules only if you have delete access to the rules.

Working with variant rules and constraints

Variant rules and constraints

Variability can be complex to understand and maintain. Offering variability to the marketplace is critical to product success.

Configurator rules or constraints allow you to specify which features and in which combination these features are allowed for a given product, a valid configuration, or a customer order. Grid and table-based interactions allow you to efficiently create, sort, and filter large sets of data. You can work in the icon-based grid view while the system constructs the Boolean expressions.

Marketing and sales organizations typically use variant rules to specify the allowed combinations of features for the product. However, such configurations are limited to those that the design engineer permitted during creation of the variant data on the structure.

To configure a particular variant of an assembly or product, you set a variant rule. The variant rule is a group of families and features such as **color = red**, **material = cotton**.

In addition to configurator rules, you can create more complex free-form SMT-based rules using the SMT Lib scripting language. Free-form rules do not contain **Subject** and **Condition** expressions. Instead, you can build a complex expression. A free-form rule is created and modified in a multiline editor.

You can save it as a configuration that you can recall, as needed. Variant rule objects store the following information:

- Properties such as ID, name, and description.
- Expressions (user and system selections).
- User session information such as revision rule, effectivity, and rule date.
- Solve profile information such as validation mode and validation severity.

You can also create variant criteria which is a revisable subtype of a variant rule. You can release, status, and revise variant criteria. You revise variant criteria by selecting **File→Revise**. Once revised, the applicable revision is shown in the **Saved Variant Rules** view as per the revision rule set in the **Variability Explorer** view. You can search for variant criteria revisions by using the out of the box search in My Teamcenter.

You can also create a copy of a variant criteria by selecting **File→Save As**. You can find the newly created variant criteria object with the copied variant criteria information in the **Newstuff** folder.

> Note:
>
> The **Save As** option for variant rule and variant criteria is currently not supported in content applications.

Your administrator controls if you create variant rules or variant criteria using the **Cfg0CreateVariantRuleType** preference. The preference can be set to either **VariantRule** or **Cfg0VariantCriteria** which allows you to create either variant rules or variant criteria, respectively. By default, it is set to **Cfg0VariantCriteria**. If the value is not set, you see an error message notifying you that no valid variant rule is found in the system.

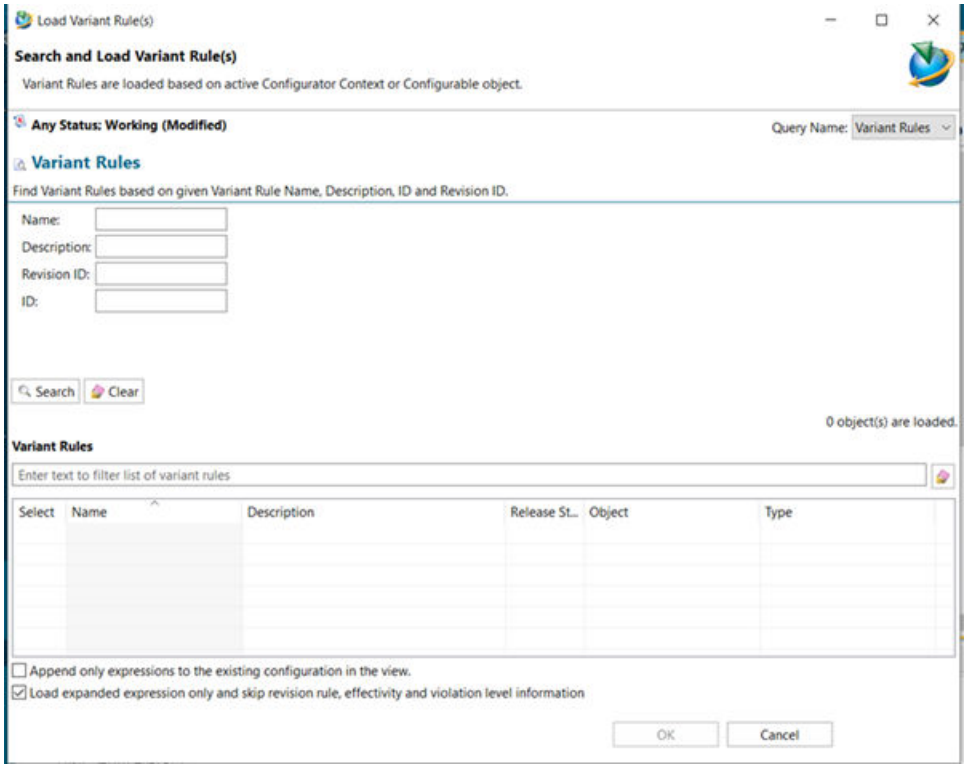
Variant rule objects may exist at the product level, where they are managed by Product Configurator; they may also be managed by the product model (for example, the product design). In both cases, the variant rules are attached to the managing object with relations that define their scope.

> Note:
>
> Your administrator can create company-specific subtypes of any existing configurator rules in Business Modeler IDE.

Retrieving saved variant rules objects

Variant rule objects, such as variant rules or variant criteria, store variant configuration criteria and optional configurator validation records. In addition, variant rules store the user session information. Variant rule objects may exist at the product level, where they are managed by Product Configurator; they may also be managed by the product model (for example, the product design). In both cases, the variant rules are attached to the managing object with relations that define their scope.

In the **Variant Configuration** view, click **Load Variant Rule(s)** to search and filter variant rules or variant criteria using your defined criteria in this dialog box.



You can specify the revision rule in the **Load Variant Rule(s)** dialog box to locate the required revision of your variant criteria.

Your administrator controls which queries and search criteria are displayed in the **Load Variant Rule(s)** dialog box using the **PCA_VariantRuleSearch_SavedQueries** preference. This preference sets the default value as **Variant Rules**, which is a generic saved query used to perform a search for variant rules. You can define your own saved query for variant rules and add its name to the **PCA_VariantRuleSearch_SavedQueries** preference.

If your administrator adds configurator context to the search criteria, the system locates only variant rules that are shared with the current and active configurator context in the **Variant Configuration** view.

The **Load expanded expression only and skip revision rule, effectivity and violation level information** check box is governed by the **Cfg0LoadExpressionInfoForVR** preference. By default it is set to **false**.

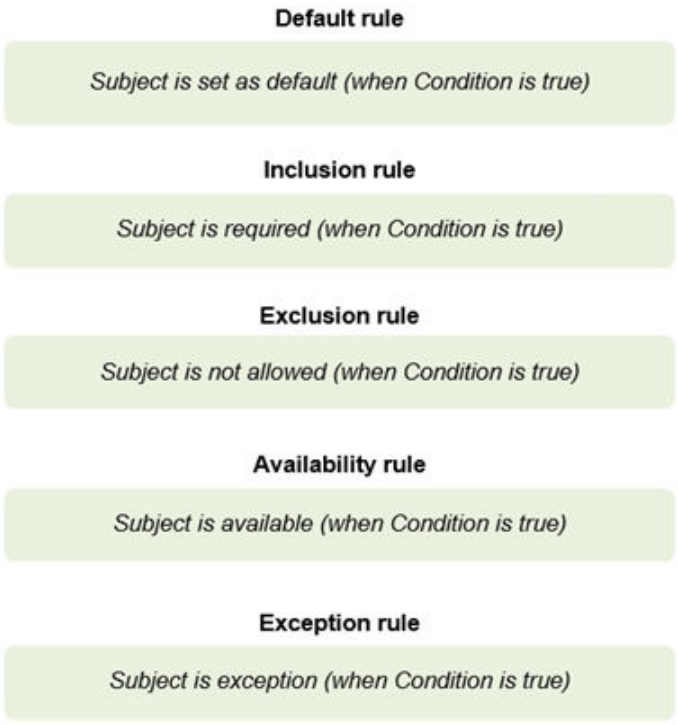
When an SVR or variant criteria is loaded:

- If this check box is selected, it loads only the expressions without the session information and as **Custom Configuration**.
- If this check box is cleared, it loads the expressions along with the session information. Additionally, the name of the SVR is displayed in the header of the **Variant Configuration** view.

Authoring variant constraints and validating configurations

Types of configurator rules

Configurator rules reference product models through an applicability expression. Product models behave as features in this expression.



The configurator rules are defined as follows:

- **Default rules** specify default features to use during configuration. Default rules are workspace objects with their own names, description, and owner. Default rules may have applicability conditions that allow you to differentiate between product models when they are applied.

  Only one literal feature is allowed in the **Subject** section.

- *Inclusion* rules are met when the condition is satisfied. Teamcenter modifies the configuration expression to enforce the constraint.

- *Exclusion* rules are met when the condition is not satisfied. Teamcenter does not attempt to modify the configuration expression to satisfy the exclusion rule. It does display a message when the exclusion rule is violated.

- *Availability rules* , like exclusion rules, do not attempt to modify the configuration expression to satisfy the availability rule. Availability rules are only used in validation requests, while inclusion rules are used in validation and translation requests.

  Only one feature (model or non-model) is allowed in the **Subject** section.

  Availability rules use normal business language which allows you to write your configuration validation statements in a user friendly manner. For example, the **Cold Weather** package is available for **Nordic** models if sufficient electrical power is available, such as a large battery and a fuel cell engine.

  > Note:
  >
  > To create availability rules, you must have an appropriate license level. If the license is not available, the system generates an error during the creation of this rule.

- *Exception* rules allow you to exclude a specific feature within a dynamic family from being subject to a mutual exclusive restriction. They help you define exception conditions for a dynamic family.

  For example, the **Black** exterior color is an exception for **X5** car models only.

  > Note:
  >
  > This rule is only available if your administrator enabled the creation of dynamic families.

In addition to configurator rules, you can create more complex *free-form* SMT-based rules using the SMT Lib scripting language without the **Subject** and the **Condition** expressions. Instead, you can build a complex expression. A free-form rule is created and modified in a multiline editor.

For example, `“Always: [CAR]Weight > [TRAILER]Weight” •“If [CAR]TowingPackage = true: [CAR]Weight > [TRAILER]Weight”`.

You can view the full constraint expression, in a format that is most convenient for you, using one of the following:

- Icons in the grid view
- Boolean expression shown at the bottom of the grid view
- Standard Teamcenter summary view
- **String Representation** column in the **Configurator Rules** view

Understanding configurator rules

You can use Product Configurator to write Boolean (**default, exclusion, exception, inclusion, availability**) and free-form rules.

Users of other applications can request Teamcenter to validate their variant criteria against these constraints. You can qualify constraints for a specific set of families.

- **Exclusion rules**

  Exclusion rules describe a disallowed condition.

  The following exclusive constraint disallows a Bluetooth device for cars with the AWD drive

  Example 1:

  ```
  Report message "Your Message" with severity "Your Severity" when
  violated.
  "[CAR]Bluetooth = true" is not allowed when "[CAR]Drive = AWD".
  ```

  Example 2:

  ```
  Report message "Your Message" with severity "Your Severity" when
  violated.
  "[CAR]Airbag = 6 AND [CAR]HillAssist = true" is not allowed when
  "[CAR]Drive = AWD".
  ```

  The second example means the system does not allow the combination of the function **Airbag = 6** and **HillAssist** with the **AWD** drive. However, users can choose **Airbag = 6** or **HillAssist** with **AWD** drive.

  The system does not present *excluded features* to users for selection during *guided validation*.

  Exclusion rules may also be constructed without an applicability condition.

  ```
  Report message "Your Message" with severity "Your Severity" when
  violated.
  "Subject" is not allowed.
  ```

- **Inclusion rules**

  Inclusion rules describe a required condition.

  Inclusion constraints express *Do* or *Must Do* conditions. They contrast with exclusive rules that express *Don't* conditions. If an inclusive constraint says "If Exterior=Black → Interior=Anthracite", the

system forces anthracite interiors for black exteriors. Conversely, if an exclusive constraint says “If Exterior=Black → Interior=Anthracite”, the system disallows anthracite interiors for black exteriors.

The following example of an inclusive constraint disallows the use of a Bluetooth device for cars with the **AWD** drive.

```
Report message "Your Message" with severity "Your Severity" when
violated.
      "[CAR]Bluetooth = false" is required when "[CAR]Drive = AWD".
```

Inclusion rules may also be constructed without an applicability condition.

```
Report message "Your Message" with severity "Your Severity" when
violated.
"Subject" is required.
```

- **Availability rules**

  Availability rules define conditional restrictions on which features should or should not be available. *All* availability rules for a given **Subject** condition jointly define the availability for the given subject, such as for a given **Subject** one of the availability rules must apply. For all other rule types it is necessary that all rules must apply.

  An availability rule is a convenient way to represent *May Do* conditions. If an availability rule says “Interior=Anthracite is available ? If Exterior=Black”, anthracite interiors are available for black exteriors.

  For example, an availability rule may assert that a feature is available only when certain features are selected or may assert that feature is available only when certain other features are not selected.

  Subject of an availability rule must be precise:

  - Comparison operators <, >, <=, >= are not allowed.
  - Only one feature with an **equalTo** operator is allowed for enumerated families.
  - Either **equalTo** and **notEqualTo** operators are allowed for Boolean families.

  A **Subject** expression must be a literal expression, for example **Transmission= Automatic**, **AutomaticBreakingSystem = AutomaticBreakingSystem**, **AutomaticBreakingSystem ! = AutomaticBreakingSystem**. Compound expressions are *not* allowed, such as **Transmission= Automatic AND AutomaticBreakingSystem = AutomaticBreakingSystem**.

  You do not have to specify a model when constructing availability rules. You can have a model or other features in the availability statement. For example, you can make an engine available, or you can make an engine available to transmission, or you can make an engine available to transmission for a specific model.

```
Report message "Enter Message Here" with severity "Error" when
violated.
"[CAR]Bluetooth= true" is available when "[CAR]Region = Any".
```

Currently, the system evaluates all availability rules as if they had severity level **Error**. Siemens Industry Software Inc. recommends to keep all availability rules at severity level **Error** until the future release with an explicit support of multiple severity levels for availability rules.

Summary models, feature summaries, and product lines are always available by default. There is no need to create availability rules for these object types. Your administrator can change the list of objects exempt from the availability check by modifying the **Cfg0ExemptAvailabilityChecks** business constant.

If existing constraints reduce the available variability of the family to a single feature, requests for available variability only return this feature. However, automatic variant criteria completion does not automatically assign the remaining feature, even if the family is marked as mandatory, unless the feature is forced on by an include rule.

However, the system behavior for availability rules is to combine all availability rules for the same subject with the Boolean operator OR. An individual availability rule can therefore not be violated in the current system. Therefore, violation messages of availability rules are not displayed. When none of the availability rules is satisfied, the system automatically generates a violation error message with severity **Error**.

- **Exception rules**

  Exception rules define a condition under which a standalone feature is NOT considered to be a member of a dynamic family.

  ```
  Report message "Your Message" with severity "Your Severity" when
  violated."
  [[Teamcenter]Seats]Taxi" is an exception for "Dynamic Family Thread"
  [Teamcenter]Seats when "[026025]Models = X5".
  ```

  Exception rules may define a model or effectivity criteria. If an exception rule applies a standalone feature, it may remain a member of allocated dynamic families, but it is no longer mutually exclusive with other standalone features in this dynamic family. Exception rule is not relevant for multiselect dynamic families.

  > Note:
  >
  > Exception rules are only available if your administrator enabled the creation of dynamic families.

- **Free-form**

  Free-form rules are more complex free-form SMT-based rules. You create them using the SMT Lib scripting language.

A free-form rule is created and modified in a multiline editor.

```
“Always: [CAR]Weight > [TRAILER]Weight” •“If [CAR]TowingPackage = true:
[CAR]Weight > [TRAILER]Weight”
```

By default, the system converts inclusion and exclusion rules into a read-only free-form expression that you can view in the **Free-Form Expression** column. However, the default, exception, and availability rules are not displayed. You can modify the SMT expression for the free-form rules only.

Severity levels

For each constraint, you can set one of three severity levels values — information, warning, or error. Teamcenter displays a message with the appropriate severity value if a constraint condition violation occurs. A message occurs only after Teamcenter considers every possible way of satisfying the variant configuration criteria and determines whether the remaining variant configuration criteria allow you to avoid a constraint condition violation.

Constraint rule messages

The **Message** attribute of every constraint rule stores a localizable text message of **Enter message here** which is displayed when this rule is violated. You can change the **Enter message here** before the initial save, if you have write permission for the constraint rule.

> Note:
>
> Constraint messages may be localized so that they are displayed in the user’s current locale. Currently, you cannot localize or change the system-generated availability rule violation messages. However, you can still edit and localize text in the **Configurator Rules** view.

How constraint rules map to Boolean operators

In Product Configurator on Rich Client — Usage, constraints map to Boolean operators. In general, 16 **Boolean operators exist**. The most commonly used operators are AND, OR, and NOT. Some of the other operators are as follows:

Operator	Meaning	Notation	AND/OR/NOT equivalent
Implication	if, then only if	X → Y	Y or not X
Equivalence	if and only if	X ≡ Y	X and Y, or not X and not Y
Exclusive Or	either, or	X≠ Y	X and not Y, or not X and Y

Model memberships	These define equivalence constraints. You can select the product model in the **Variant Configuration** view, or all of its members. From a constraint evaluation perspective, both selections are

	equivalent and are denoted as **Model ≡ AND(V1 ,..., Vn)**, where **V1**, and **Vn** are model members. The system-generated equivalence constraint has error severity. When the constraint is violated, a system-generated message is shown that is based on TextServer key "k_equivalence_error_message" and "k_product_model", for example, “The equivalence constraint between the object "[[SPLM]Models]LXI" of type "Product Model" and its members cannot be satisfied.” in the English locale. An invalid combination of members invalidates the product model. Validating a configuration for this product model fails with the error severity showing a message that is based on TextServer key "k_member_configured_error_message" and the "Display Name" for the product model, for example, “Members of "LXI" are not configured”.
Product line memberships	These define equivalence constraints. Product line memberships differ from model memberships in that more than one feature of the same family can be a member of a given product line. Member features in the same family are ORed. The results are then ANDed. This is similar to how the **Variant Expression Editor** view combines selections in the same column for the same family with an OR function. Results from all such OR functions in the same column are combined with an AND function. The following formula refers to this logic as `EDITOR`, for example, **Product Line≡ EDITOR(V1 ,..., Vn)**, where **V1**, and **Vn** are product line members. The system-generated equivalence constraint has error severity. When the constraint is violated, a system-generated message is shown that is based on TextServer key “k_equivalence_error_message” and “k_product_line”, for example, “The equivalence constraint between the object "[[SPLM]Line]Cargo" of type "Product Line" and its members cannot be satisfied.” in the English locale. An invalid combination of members invalidates the product line, which cascades to all child product lines and product models. Validating a configuration for such a product line or product model fails with the error severity showing a message that is based on TextServer key “k_member_configured_error_message” and the “Display Name” for the product line, for example, “Members of "Cargo" are not configured”.
Feature summaries	These define equivalence constraints. Feature summaries differ from equivalence constraints in that a feature summary is equivalent to an exclusive OR combination of its members, denoted as **Summary ≡ XOR(V1 ,..., Vn)**.
Feature packages	These define implication constraints. A package can be expanded into an AND combination of its members. However, selecting all members does not automatically also set the package as

	well. The difference between implication and equivalence is that implications only work in one direction, denoted as **Package → AND(V1 ,..., Vn)**. The system-generated implication constraint has error severity. When the constraint is violated, a system-generated message is shown that is based on TextServer key “k_package_expansion_error_message”, the feature, and “k_package_option_value_category”, for example, “The implication constraint between the object "” [[SPLM]Packages]Pkg1"” of type "”Feature Package"” and its members cannot be satisfied” in the English locale. An invalid combination of Package members invalidates this Feature package. Validating a configuration for this Feature package fails with error severity showing a message that is based on TextServer key “k_member_configured_error_message” and the “Display Name” for the feature package, for example, “Members of "”Pkg1"” are not configured”.
Include rules	These define implication constraints. The **Condition** section **A** implies the **Subject** section **S**, denoted as **A→S**. If the **Condition** section is empty, it is interpreted as **TRUE** or **Always**. The effectivity condition **E** of the include rule is added to the **Applicability** section. If the effectivity condition is empty, it is interpreted as **TRUE** or **Always**, and denoted as **AND(A, E) → S**.
Default rules	The behavior of a default rule depends on its property **cfg0UseOnlyFwdPropagation** (Use only Forward Propagation in the English locale). If the property has a value of **FALSE**, the default rule behaves just like an include rule (see above). See **About default rules** for more detail about the difference and a recommendation. Default rules with a property value of **cfg0UseOnlyFwdPropagation=TRUE** encode a normal default logic with the following syntax: $$Default\ Rule = \left\{ \frac{A\ :S}{S} \right\}$$ If the current configuration settings require the **Condition** section **A**, and the **Justification** given in the **Subject** section **S** is consistent with this configuration, that is, **S** is satisfiable, then the **Conclusion S** can be drawn (see **Default logic**). **Error**, **Warning**, and **Information** severity constraints have a **higher severity level than the default rules do**. The evaluation

	of these severity constraints can therefore restrict the **Condition** section **A** or the **Justification** in the **Subject** section **S** of a default rule. The former inactivates a default rule, while the latter overrules it. Either way, such a default rule would not conclude its **Subject** section **S**, and would do so in a silent way. If the **Condition** section is empty, it is interpreted as **TRUE** or **Always**. The effectivity condition **E** of the default rule is added to the **Applicability** section. If the effectivity condition is empty, it is interpreted as **TRUE** or **Always**.
Exclude rules	These define implication constraints, but the **Subject** section **S** is implicitly negated: The **Condition** section **A** excludes the **Subject** section **S**. If the **Condition** section is empty it is interpreted as **TRUE** or **Always**. This is denoted as **A→ not S**. The effectivity condition **E** of the exclude rule is added to the **Applicability** section. If the effectivity condition is empty, it is interpreted as **TRUE** or **Always**, and denoted as **AND(A, E) → not S**.
Availability rules	The set of availability rules for a given **Subject** jointly define a single implication constraint for negative biased configurator contexts. The **Subject** section **S** implies that at least one of the **Condition** sections **A** is satisfied, denoted as **S →OR(A1 ,..., An)** The effectivity condition **E** of each availability rule is added to the respective **Applicability** section. If the effectivity condition is empty it is interpreted as **TRUE** or **Always**, denoted as **S →OR(AND(A1, E1), AND(A2,TRUE), AND(A3, E3), ...)**. The system-generated implication constraint that represents the combined set of availability rules has error severity. When the constraint is violated, a system-generated message is shown that is based on TextServer key “k_option_value_not_available_error_message” and the feature, for example, “The Feature “[[SPLM]Version]V1XXX” is not available.” in the English locale. Note: The **Subject** section **S** is on the left side of the implication operator → (denoted using a right arrow symbol). In all other constraints, the **Subject** section **S** is on the right hand side of the implication operator.
Allocation constraints	The system generates a constraint with error severity that asserts a valid feature selection of feature values **V1, ..., Vn**, in the feature family **F**, denoted as **OR(F=V1 ,..., F=Vn)**.

	If the feature family is optional the “NONE” value is automatically included in the list of choices, for example, **OR(F=V1 ,... , F=Vn, F=NONE)**. For more information about optional feature families, see **Distinguishing between mandatory and optional families**. If the family defines minimum or maximum value the system-generated constraint asserts the domain of value: **AND(F>=min, F<=max, OR(F=V1, ..., F=Vn, F=NONE))**. The constraint is violated if no valid choice exists. In that case, a system-generated message is shown that is based on TextServer key “k_variant_criteria_outside” and the “Display Name” for the feature family **F**, for example, “The configuration criteria are outside the allowed value range for option “F”” in the English locale.
Effectivity constraints	The net effectivity of a feature is the intersection between the effectivity of the feature revision and its allocation revision. For more detail see **Applying effectivity on your configurator data**. The system generates an effectivity implication constraint with error severity for every feature or product model **V** that has a non-empty net effectivity **E**, denoted as **V → E**. Validating a configuration for which the effectivity implication constraint cannot be satisfied displays a system-generated message that is based on TextServer key “k_variant_criteria_outside” and the “Display Name” for the feature or product model **V**, for example “The configuration criteria are outside the allowed value range for option “V”” in the English locale.

For information about localizing constraint messages see **Understanding configurator rules**.

> Note:
>
> All localization files are located in **$TC_MSG_ROOT**. For more information about Teamcenter's TextServer for localizable messages, see Understanding the TextServer.

Understanding severity levels in configurator rules

Configurator constraint rules, such as inclusion or exclusion rules, specify a severity level. Constraint rules might exist with multiple severity levels. In the extreme case, high-severity constraint rules may contradict low-severity constraint rules or may define exceptions.

As a BOM engineer, you can assign the severity levels **Error**, **Warning**, or **Information** for configurator rules.

> Note:
>
> The severity level for the validations cannot be specified in the **Variability Explorer**, **Variant Expression Editor**, and **Configurator Rules** views.

During the validation process, if there are multiple rules with conflicting severity levels, **the system validates and enforces the rules** based on the following severity ranking:

1. **Error** (highest severity)
2. **Warning**
3. **Information**
4. **Default** (lowest severity)

Modes and severity levels

You can use the **Order mode or the Overlay mode to configure and analyze the variant configuration**.

	Order mode	Overlay mode
Validate	Important rules are considered for both modes.	
Expand	All rules, important and unimportant, are considered.	Only important rules are considered.

The user who authors rules decides the severity level. When you validate or expand the configuration, you can decide what counts as *important*. If **Warning** is selected, then the severity rules **Warning** and **Error** are considered important and others are not.

In the **Overlay** mode, when you validate and expand the configuration, the system uses every rule that is important. Rules that are not important are assigned the lowest priority. For example, if the **Warning** severity level is selected, every rule above the selected level, that is, those at the **Error** or **Warning** severity level, are considered. The rules below the selected level, that is, those at the **Information** or **Default** severity level are ignored.

In the same scenario, in the **Order** mode, when you validate and expand the configuration, if you violate any rule above the **Warning** severity level, the system immediately displays an error.

When none of the rules above a certain severity level is violated, the system considers the rules below the threshold severity level. When you validate and expand the configuration in the **Overlay** mode, the system does not look at the rules below the severity level. However, it does in the **Order** mode. If the rules are passed or satisfied, they pick up the side effects of the rules and if they fail, they are dropped silently.

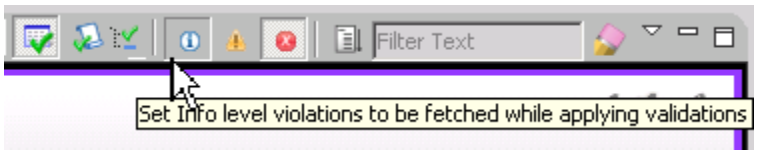
The **AND/OR/NOT** equivalent of an include rule **A Includes S (A→S)** is **Not A Or S**. For more information, see **How constraint rules map to Boolean operators**. *An information severity include rule* **A Includes S (A→S)** is therefore violated if neither **Not A** nor **S** are satisfiable. For example, an information severity include rule **""TowingPackage = FALSE" is required when "Model = Coupé""** returns an information severity validation failure for the configuration input **"Model = Coupé AND TowingPackage = TRUE"** when validating the severity level **Information**. Expanding the configuration input **"Model = Coupé"** in **Order** mode or in **Overlay** mode with severity level **Information**, expands the configuration to **"Model = Coupé & TowingPackage = FALSE"**. Expanding a configuration **"TowingPackage = TRUE"** logically expands the configuration to **"Model != Coupé & TowingPackage = TRUE"**. However, Teamcenter does not expand negative selections such as **"Model != Coupé"**. You might, therefore, not see all expand effects in the **Variant Configuration** view but a constraint-based content solve would filter out the **Coupé** product content. For more information, see Configure a structure by setting option values. If there were only two models to choose from, for example, **"Model = Pickup"** or **"Model = Coupé"**, the expand effect is displayed in the **Variant Configuration** view as **"Model = Pickup & TowingPackage = TRUE"** because the **Pickup** model is the only non-Coupé model left.

In the **Order** mode, the system uses default rules to expand the configuration beyond the severity constraints rules **Error**, **Warning**, and **Information**. This means that the side effects of the satisfied default rules are retained. Side effects occur when features specified on input or added as a result of another constraint cause a default rule to conclude its subject. Building on the example above, a default rule such as **The default configuration is "TowingPackage = FALSE" when "Model = Coupé"** does not return a validation failure. Expanding the configuration input **"Model = Coupé"** with default rules (for example, when expanding in the **Order** mode), the system expands the configuration to **"Model = Coupé & TowingPackage = FALSE"**. Expanding a configuration **"TowingPackage = TRUE"** is further not possible because neither is the applicability condition **"Model = Coupé"** of the default rule required by the configuration nor is its justification **"TowingPackage = FALSE"** consistent with the input configuration.

If a constraint set contains both an **Information** severity rule **""TowingPackage = FALSE" is required when "Model = Coupé""** as well as a **Default** rule **"The default configuration is "TowingPackage = "FALSE" when "Model = Coupé""** the default rule is rendered redundant and does not contribute to a Configurator response.

When you are validating the configuration in the **Variant Configuration** view, the system reports the constraint rule violations as follows.

- Validate with the **Information** ⓘ severity level in rich client



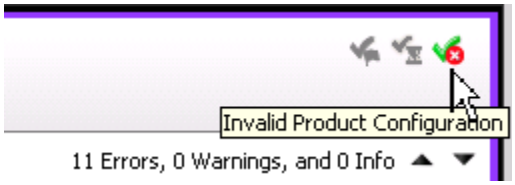
The system reports all constraint rule violations with the severity level as **Information** or higher.

Teamcenter ships with a configuration profile definition for the **Overlay** mode that does not consider constraint rules with a severity that is lower than the input severity. Because the severity of default

rules is considered lower than the severity of **Information**, **Default** rules are not applied in the **Overlay** mode.

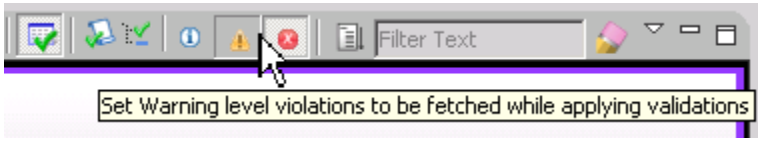
Users may want to use the **Overlay** mode with an incomplete configuration and apply the most used (defaulted) features to reduce the content for the first round of validation. In such cases, they can create a custom validation mode.

Validating a configuration that violates constraint rules with the **Information** severity level returns a validation failure. This is indicated by the validation icons at the top right corner as follows:



- Valid and complete product configuration
- Valid and incomplete product configuration
- Invalid product configuration

- Validate with the **Warning** severity level in rich client



The system reports all constraint rule violations having the severity level **Warning** or higher. Violations with lower severity are not reported.

Validating a configuration that violates constraint rules with the severity level **Warning** returns a validation failure. This is indicated by the validation icons in the top right corner.

In the **Order** mode, the side effects of satisfied constraint rules with a lower severity are retained. All violating constraint rules with a lower severity are dropped along with their side effects.

In the **Overlay** mode, a severity lower than the input severity is not considered. If the input severity is **Warning**, both the **Information** and the **Default** severities are not considered as they are lower than the **Warning** severity.

- Validate with the **Error** severity level in rich client

By default, the system reports all constraint rule violations with the severity level **Error**. Violations with lower severity are not reported.

Validating a configuration that violates constraint rules with the severity level **Error** returns a validation failure. This is indicated by the validation icons in the top right corner.

In the **Order** mode, the side effects of satisfied constraint rules with a lower severity are retained. All violating constraint rules with a lower severity are dropped along with their side effects.

In the **Overlay** mode, a severity lower than the input severity is not considered. When the input severity is set to **Error**, violations with other severity levels are not reported.

Note:

Default rules may have side effects that involve higher severity rules in a domino effect. Side effects occur because all constraints, including defaults, are evaluated simultaneously. Therefore, **Subject** in a default rule could satisfy the applicability of an include rule and can enforce its **Subject** condition. It may create a dependency chain.

In the **Order** mode, the side effects of constraint rules may propagate along a dependency chain. These side effects propagate with the original severity (that resulted in the side effect) even if they are subsequently involved with higher severity constraint rules later in the dependency chain. All these side effects disappear when the dependency chain is broken, such as when a default selection is overwritten.

About default rules

As a configurator administrator, you can use Product Configurator to create default variant rules. Default rules can improve usability because they help users form a complete configuration of a product that is ready to be validated, tested, manufactured, or to be purchased quickly. A default rule tries to anticipate common configuration settings that meet most users' preferences.

The response of a system with default rules depends on the input configuration criteria. The same set of default rules might derive different default configurations for different input criteria. For example, for a Boolean choice **A=TRUE** and **A=FALSE**, the system might have five default rules, of which three derive **A=TRUE** for certain configuration scenarios, while two derive **A=FALSE** for other scenarios. There could be input configuration criteria, for which the system derives **A=TRUE**, while other configuration criteria derive **A=FALSE**. Again other configuration criteria might not derive a value at all so that the Boolean choice remains unset until the user refines the input criteria. In an interactive configuration, the BOM engineer may choose to override the defaults. If instead of a default, the configurator administrator wants a feature to be enforced and prevent it from being overridden, use an inclusion rule.

You can control the system behavior with respect to default rules using preference settings or default rule properties.

Using preference settings:

- **PCA_solver_profiles** (Active Workspace-specific)
- **PCA_Default_Solve_Mode** (Active Workspace-specific)
- **Cfg0ApplyConstraints**
- **Cfg0DefaultRuleSortSequence**

Using default rule properties:

- **cfg0UseOnlyFwdPropagation** (Use only Forward Propagation in the English locale)

> Note:
>
> Siemens Digital Industries Software recommends setting the **cfg0UseOnlyFwdPropagation=TRUE** preference for all default rules. This preference is not available by default, so you must create it.

Default rules that are in conflict with the user input or constraint rules with a higher severity must be suppressed temporarily when expanding the configuration. Teamcenter uses a ranking system to reduce the number of conflicting default rules that must be suppressed temporarily when expanding a configuration. This ranking system is based on two strategies:

1. Logical nearness to input criteria based on **entailment** (see **Example: Expand the configuration with default rules** for an example of how the system resolves conflicts based on this strategy.)

2. Manual sequencing (see the **impact of default rule sequence numbers**)

When a new default rule is created, the default sequence number (**cfg0Sequence**) is usually **0**, that is, it is the same for all default rules. When multiple default rules with the same sequence number are involved in a conflict, the system ranks default rules higher if input configuration criteria entail their condition. It is possible that an entailed default rule might in turn entail some of the other default rules that are involved in the same conflict. The default rule that is entailed last is ranked the lowest. The system, therefore, prefers to suppress the last default rule in this entailment chain to resolve the conflict.

If the default strategy to resolve conflicts involving default rules does not achieve the expected results, Siemens Digital Industries Software recommends refining the condition expressions to be more specific to the engineering intent of the default rule.

If the automatic conflict resolution does not achieve the required results even after refining the conditions for some default rules, you can manually overwrite the automatic ranking strategy by assigning sequence numbers to the default rules.

Teamcenter evaluates default rules according to their sequence number. Default rules with a higher sequence number are evaluated later and therefore have a lower ranking. Those with the same

sequence number have the same ranking. The result of evaluating default rules with the same ranking does not depend on processing them in any particular order. Product Configurator follows a **declarative paradigm**. The set of constraint rules builds a structure that expresses the logic of validating configuration inputs without describing the control flow. This means that the result of evaluating default rules with the same ranking is consistent and does not depend on the sequence in which they are loaded from the database, created, changed, or *evaluated*.

If default rules are expected to behave like inclusion rules (that is, like **Boolean Implications**), at default rule severity level, you can set the **cfg0UseOnlyFwdPropagation=FALSE** property. For example, a product may have two Boolean choices **A** and **B** and a default rule **A → Not(B)**. A user selects **B** in the **Variant Configuration** view and performs an expansion of this input configuration:

Boolean choice	**Input**	**Expected expand result**	
A	*unset*	*unset*	**FALSE**
B	**TRUE**	**TRUE**	**TRUE**
Recommended value for default rule property, **cfg0UseOnlyFwdPropagation**		**TRUE**	**FALSE**
Explanation		Default rule is overridden	No conflicts exist that need to be resolved. A Boolean Implication **A → Not(B)** is logically equivalent to **B → Not(A)**

When in doubt, Siemens Digital Industries Software recommends working with property value, **cfg0UseOnlyFwdPropagation=TRUE**.

Default rules are always evaluated in the context of all higher ranked constraints. The configuration input, for example, from a saved variant rule (SVR), and *Error*, *Warning*, or *Information* severity constraints have a **higher ranking because of their severity**. Among default rules, the ranking can be enforced by their sequence number.

The impact of default rule sequence numbers is explained using the following example, where all default rules have the same sequence number:

- Model := { M1, M2 }
- Boolean Features “A” and “B”
- Single Select Family X := { x1, x2, x3 }
- Default Rule D1 (Sequence **10**): A → x1
- Default Rule D2 (Sequence **10**): B → x2

For more information on creating families and features, refer to **Defining dynamic families and standalone features**.

The **Subject** sections of the two default rules D1 and D2 are in conflict. For the configuration input of **M1 & A & B**, the system can apply either D1 or D2 but not both. As D1 and D2 have the same ranking, the system cannot further expand the input **M1 & A & B**. The user would need to choose a value in the family X in order to obtain a complete configuration.

For the configuration input **M1 & B**, the system can apply D2 but not D1. The configuration can therefore be expanded to **M1 & B & x2**. On the other hand, for the configuration input **M1 & A**, the same set of default rules would expand to **M1 & A & x1**. Neither of the two scenarios constitute a conflict. Therefore, although the expanded result does not depend on sequential rule processing, it may depend on the sequence in which the input is provided. After expanding **M1 & A** to **M1 & A & x1**, if the user also selects **B**, the previously expanded selection of x1 is revoked again because **D1** and **D2** are in conflict with each other once again.

Now let us look at what happens if the sequence of the default rule **D2** is changed from **10** to **20**:

- Model := { M1, M2 }
- Boolean Features “A” and “B”
- Single Select Family X := { x1, x2, x3 }
- Default Rule D1 (Sequence **10**): A → x1
- Default Rule D2 (Sequence **20**): B → x2

With this change, the system first evaluates all default rules with sequence **10** (in this example, only D1) in the context of the configuration input and the higher ranked constraints. In this example, there are no other constraint rules. Only the configuration input is ranked higher than D1.

For the configuration input of **M1 & A & B**, the system first applies **D1**, which expands the configuration to **M1 & A & B & x1**. When the system evaluates the default rules at sequence 20, the justification for D2 (**x2**) conflicts with the configuration. The default rule **D2**, therefore, does not contribute to the expand result. The result of expanding the configuration is **M1 & A & B & x1**.

> Note:
>
> For information on the syntax of default rules and their constituent elements *Condition*, *Justification*, and *Conclusion*, see **How constraint rules map to Boolean operators**.

For the configuration input **M1 & B**, it remains unknown whether the configuration settings match the applicability condition of **D1** as it depends on subsequent user inputs for feature A.

Note:

The absence of A in the configuration input implies that A is *unknown*. It does not mean that A is *FALSE*.

When the system advances to the default rules with sequence **20**, it is equally unknown whether the justification x2 is consistent or if it conflicts with the configuration. If the user eventually selects **Not A**, the justification then becomes consistent, and the default rule can expand the configuration input to **M1 &!A &B & x2**. If not, the justification conflicts with the configuration. This is because for the configuration input **M1 & A & B**, the higher-ranked default rule D1 would have expanded to **M1 & B & A & x1** before reaching **D2**, and this in turn conflicts with **D2's** justification **x2** in the subject condition.

The result of expanding the configuration is **M1 & B**. No default value in the family **X** is selected.

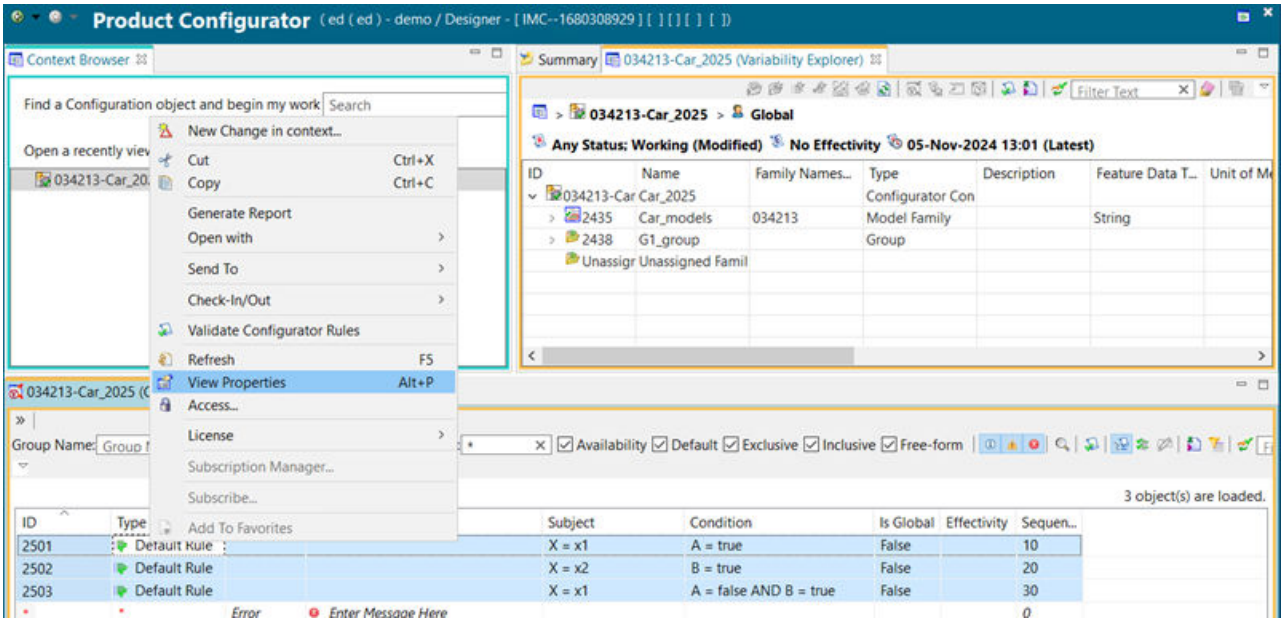
Product Configurator never configures out any relevant content and never expands any selection that is not relevant when expanding a configuration or when configuring content. It might configure more content than the absolute minimum but never less. It may expand fewer default selections than the absolute maximum but never more.

Continuing with the example, in the following scenario, one more default rule, **D3**, is added at sequence **30**:

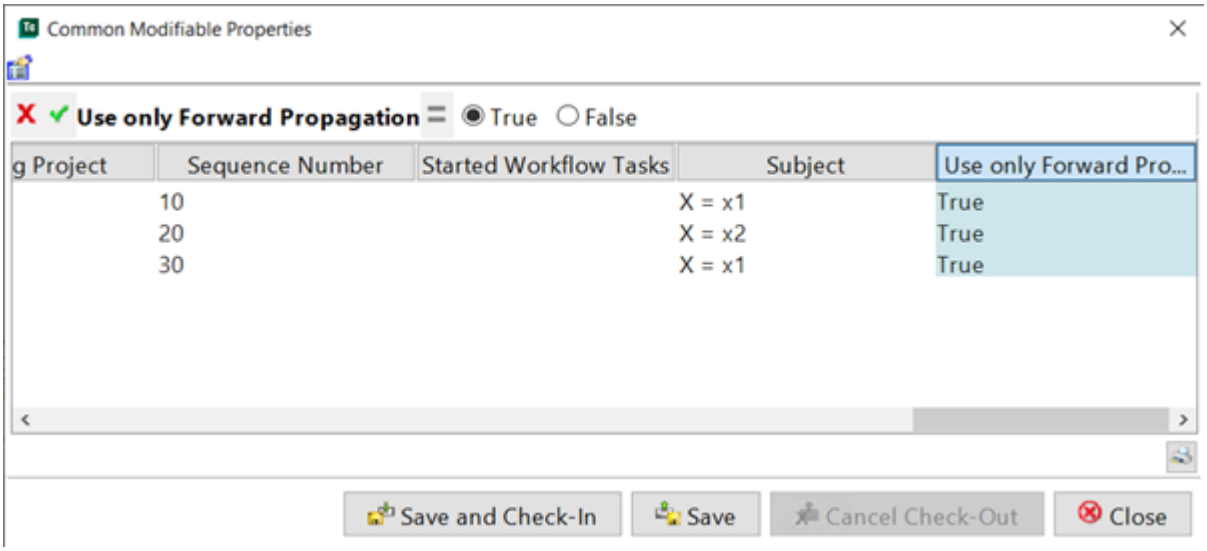
- Model := { M1, M2 }
- Boolean Features "A", "B"
- Single Select Family X := { x1, x2, x3 }
- Default Rule D1 (Sequence 10): A → x1
- Default Rule D2 (Sequence 20): B → x2
- **Default Rule D3 (Sequence 30): ! A & B → x1**

Perform the following steps:

1. Open the configurator context, click the **Open Configurator Rules view** button, select all the default rules, right-click, and choose **View Properties**.

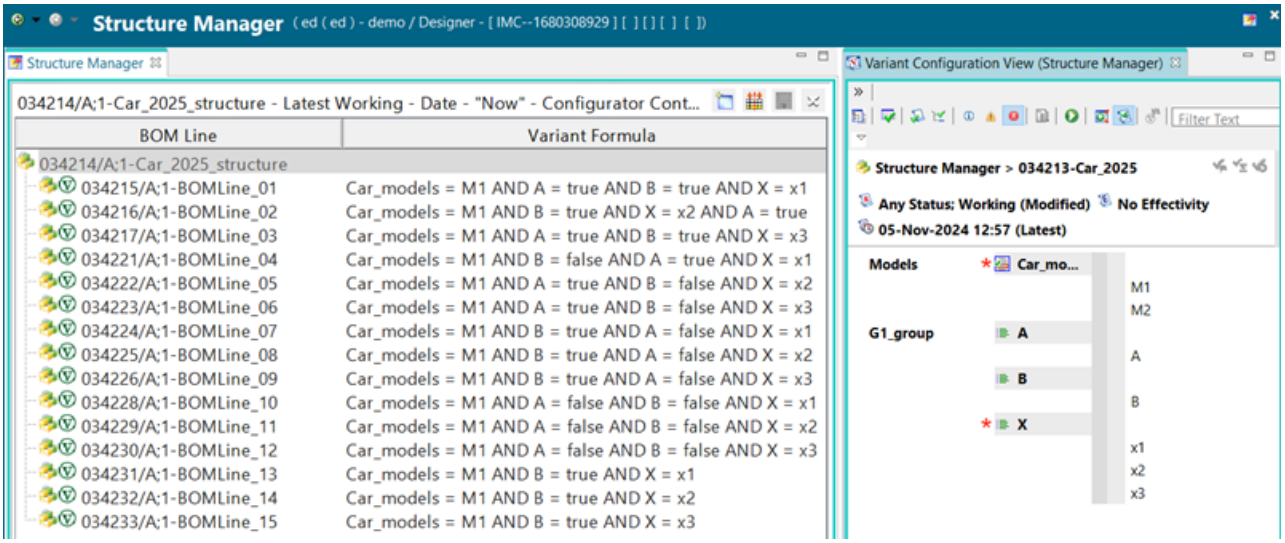


2. Choose **Check-Out and Edit**, select **Use only Forward Propagation**, and click **True**.



3. Choose **Save and Check-In**.

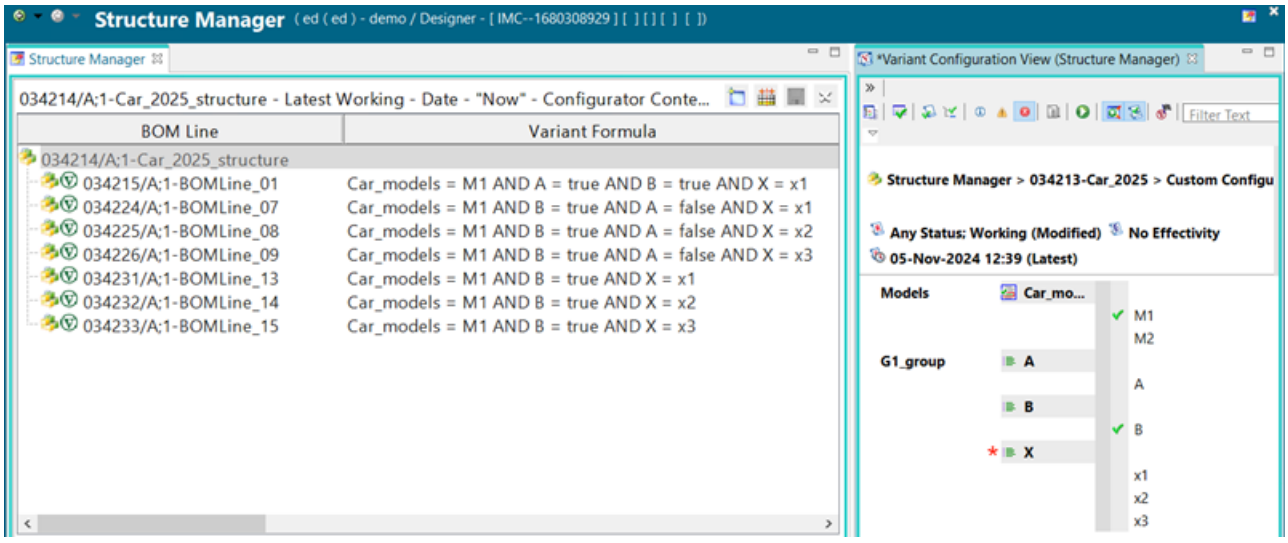
4. Open the structure in Structure Manager and open the **Variant Configuration** view.



5. Make selections for **M1** and **B** and apply constraints by selecting the **Apply Constraints toggled on**.

6. Click the **Expand** button and select the **Applies the current configuration to the associated structure** button.

In this case, the system retains the default rule **D1** but removes the default rules **D2** and **D3** when expanding a configuration or when configuring content. This guarantees that no relevant content is ever configured out, for example, content with the variant condition **!A B & x2**. In some cases, irrelevant content might not be configured out, such as content with the variant condition **!A & B & x1**. The following screenshot displays the result of a constraint-based content solve for **M1 & B** with defaults:



Each row represents a variant condition that combines the features in each cell with AND. The features that violate the highest-ranking rule are marked in red. The conditions that pass a content solve even though they are not relevant for any default are marked in purple.

You can assign a default based on multiple applicability conditions.

Example:

Default rules may be created with or without an applicability condition.

- The following example shows a default rule created with an applicability condition.

```
The default configuration is "Subject" when "Condition".

The default configuration is "B = b1"
when "(A=a1 OR A=a2) AND C=c1 AND C=c2".
```

- The following example shows a default rule created without an applicability condition.

```
The default configuration is "Subject"

The default configuration is "B = b1".
```

Example: Expand the configuration with default rules

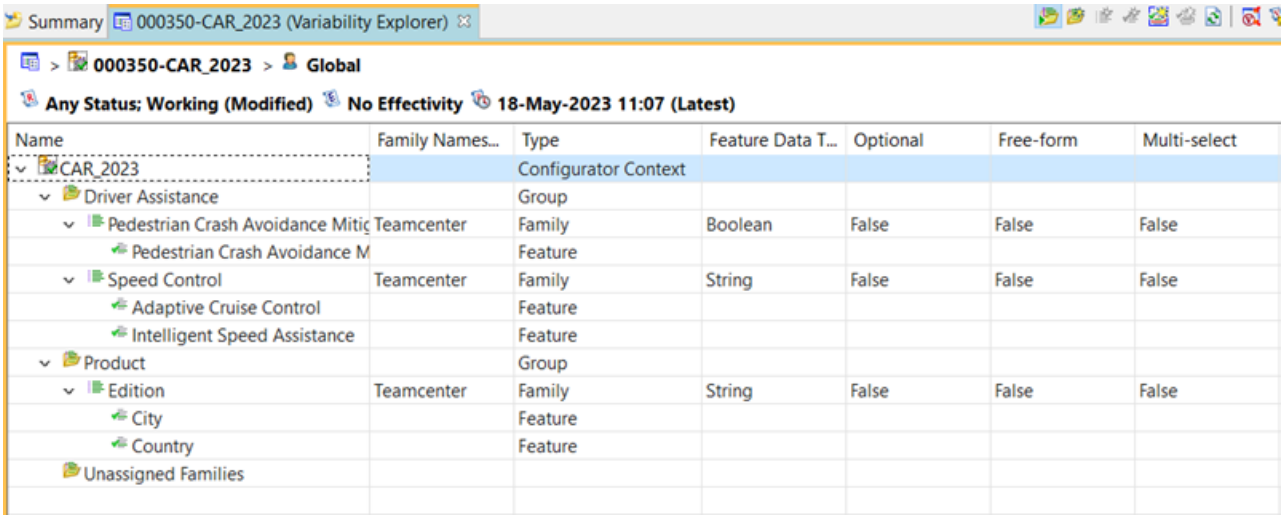
The **Variant Configuration** view expands the configuration input criteria based on a set of configurator rules. When the validation mode is set to **Order** mode, the rules also include default rules. In the manual validation mode, the criteria are expanded with the **Expand** button.

Default rules can improve usability because they help users form a complete configuration of a product quickly. A default rule tries to anticipate common configuration settings that meet most users' preferences.

Whenever the system runs default rules, there would be situations with conflicting rules. Default rules that are in conflict with the user input or constraint rules with a higher severity must be dropped when expanding the configuration. To reduce the number of conflicting default rules that must be dropped when expanding a configuration, Teamcenter evaluates default rules according to their sequence number. Because these are default rules, some of them can be dropped. The more rules the system drops, the lesser complete configuration is achieved.

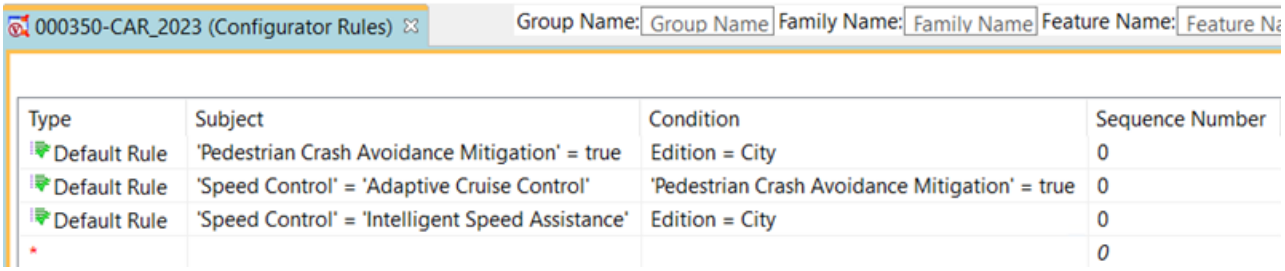
Earlier, due to some automatic conflict resolutions, the system dropped more default rules because of which there were more incomplete configurations. Now, when there are conflicts with equally ranked default rules, the system tries to keep the maximum number of rules, but with no conflicts. It tries to solve the conflict in such a way that the maximum default rules can be retained. This results in better expanded results with a lesser manual effort to manage the sequence numbers of the default rules.

1. Create a positive-biased configurator context as follows:



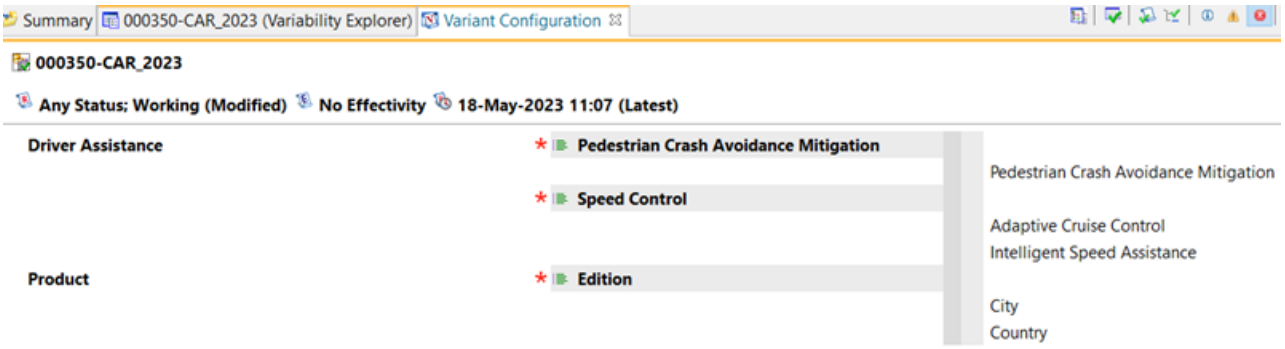
Name	Family Names...	Type	Feature Data T...	Optional	Free-form	Multi-select
CAR_2023		Configurator Context				
Driver Assistance		Group				
Pedestrian Crash Avoidance Mitic	Teamcenter	Family	Boolean	False	False	False
Pedestrian Crash Avoidance M		Feature				
Speed Control	Teamcenter	Family	String	False	False	False
Adaptive Cruise Control		Feature				
Intelligent Speed Assistance		Feature				
Product		Group				
Edition	Teamcenter	Family	String	False	False	False
City		Feature				
Country		Feature				
Unassigned Families						

2. Create the following default rules with the same sequence number, that is, **0**:



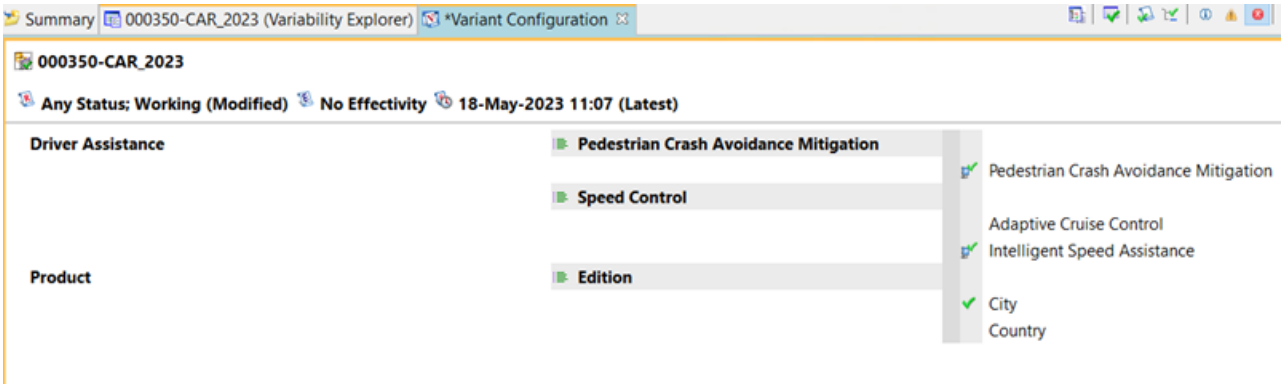
Type	Subject	Condition	Sequence Number
Default Rule	'Pedestrian Crash Avoidance Mitigation' = true	Edition = City	0
Default Rule	'Speed Control' = 'Adaptive Cruise Control'	'Pedestrian Crash Avoidance Mitigation' = true	0
Default Rule	'Speed Control' = 'Intelligent Speed Assistance'	Edition = City	0
*			*0*

3. Open the **Variant Configuration** view.



4. Choose the **Order** mode, select the **City** product edition, and click **Expand**.

   Starting Teamcenter 14.3 release, the system expands the input to a valid and complete configuration. Prior to this, the system did not select any additional default features while expanding the criteria.



About include rules

In the automotive and aerospace industries, the product configurator application helps build valid (variant configuration) orders. The product configurator defines variability through families and features, and their relationships are managed by constraints.

Customers can build a variant configuration by selecting specific features. Each feature selection triggers a set of constraints, which may add more features to the final order.

The product configurator generates orders (variant configuration) using a declarative solver[1]. This solver evaluates all constraints simultaneously, ensuring that the impact of one constraint on others is always considered. An *include constraint* is one such constraint. When include constraints enforce conflicting features, the declarative solver results in an invalid state.

Consider the following example configuration:

Variability:

Family	Features
Models	**Luxury, Economy**
Fuel Type	**Electric, Petrol**
Engine	**X40, X60, X90**
Transmission	**Automatic, Manual**

Include constraints:

Constraint ID	Constraint Type	Condition	Subject
#1	**Include**	**Luxury and Electric**	**X40**
#2	**Include**	**Luxury**	**X60**

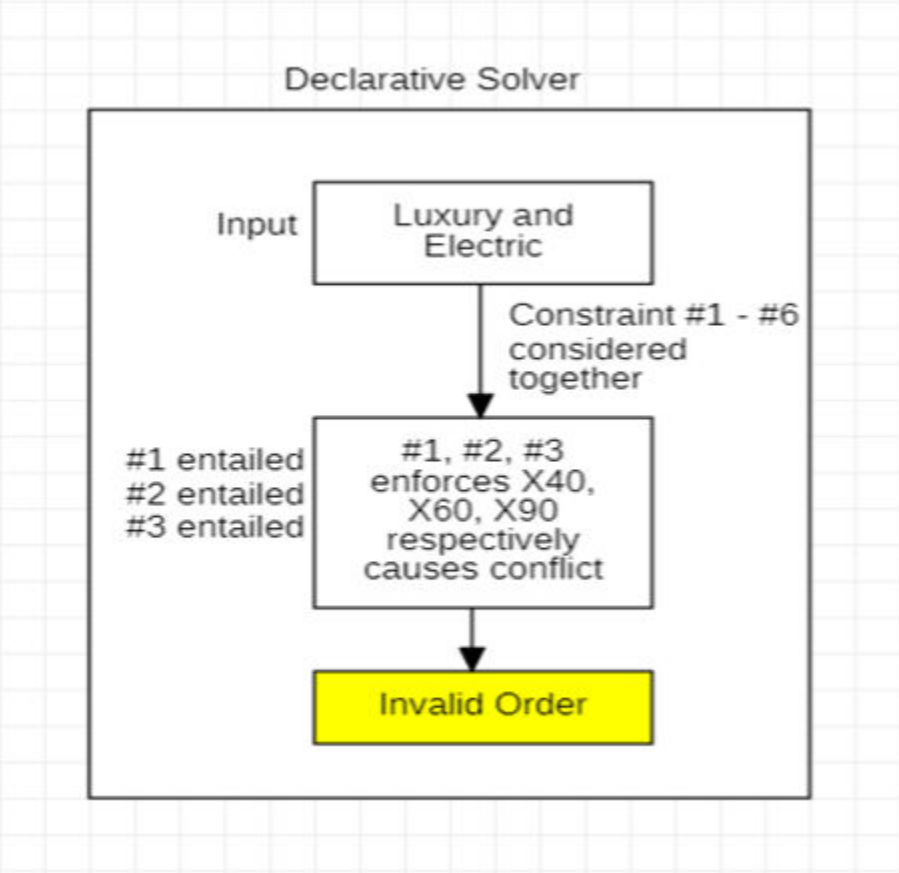
1 A declarative solver refers to a system or tool that allows users to express problems in terms of constraints without specifying the sequence or process for solving them.

Constraint ID	Constraint Type	Condition	Subject
#3	**Include**		**X90**
#4	**Include**	**Luxury and X40**	**Automatic**
#5	**Include**	**Luxury**	**Automatic**
#6	**Include**		**Manual**

Case 1:

User selection is **Luxury and Electric**

Treating include constraints as regular constraints without considering sequences causes conflicts and results in an invalid state.



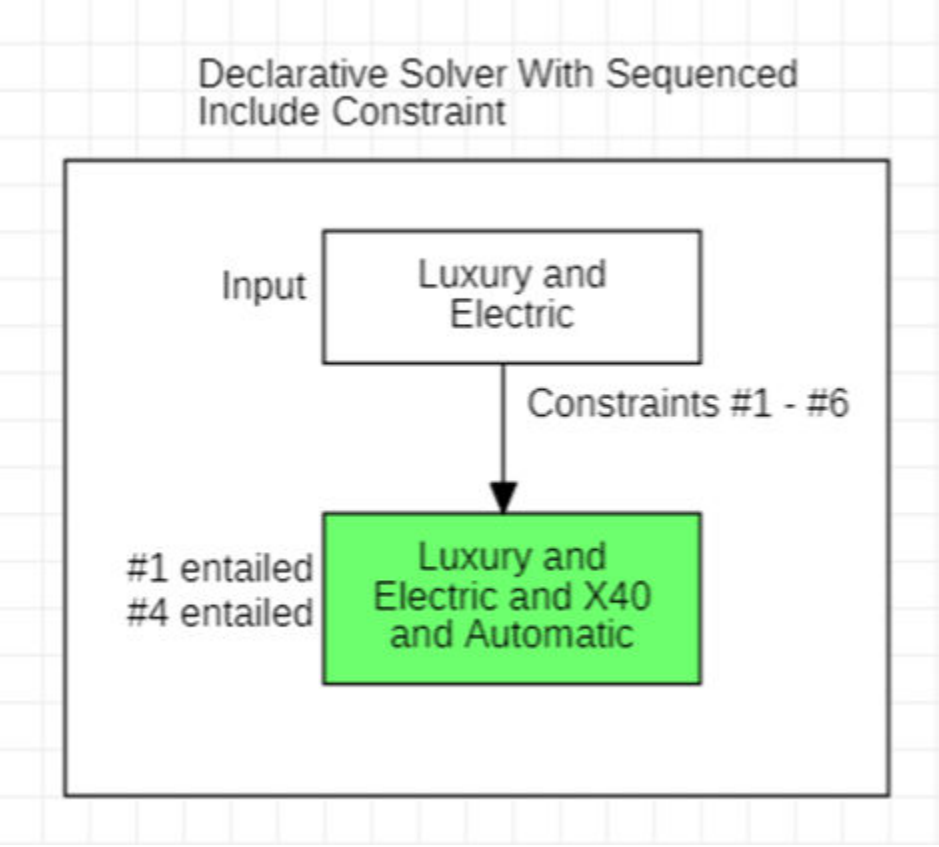
Customers familiar with an imperative solver[2] expect to build a valid order configuration using a group of sequential constraints and generate the final order without any violations. They expect the weight or priority expressed through sequences on constraints to be honored during order generation.

2 An imperative solver refers to a system that emphasizes a sequential, step-by-step procedural approach to problem-solving.

Sequenced include constraints function like regular include constraints but add *implies* behavior along with *sequences* and *sequence group*. These constraints honor sequence prioritization using a declarative solver.

Constraint ID	Constraint type	Condition	Subject	Sequence	Sequence group
#1	**Include**	**Luxury and Electric**	**X40**	**10**	**Engine Group**
#2	**Include**	**Luxury**	**X60**	**20**	
#3	**Include**		**X90**	**30**	
#4	**Include**	**Luxury and X40**	**Automatic**	**10**	**Transmission Group**
#5	**Include**	**Luxury**	**Automatic**	**20**	
#6	**Include**		**Manual**	**30**	

Expansion for case 1 with sequenced constraints: **Luxury and Electric** and **X40** and **Automatic**



Define and manage sequenced include constraints

An include constraint qualifies as a sequenced include constraint when it has the *sequence* and *sequence group* properties set.

To qualify as a sequenced include constraint, the following conditions must be met:

1. The sequence property is populated with a positive value greater than zero.

2. The sequence group property is populated with a non-empty string.

3. Sequences within a sequence group must be unique.

> Note:
>
> If an include constraint does not meet any of the above requirements, it is treated as a regular Include constraint. This includes all constraints within that sequence group, and the declarative solver treats them at the same level.
>
> The sequence property on the include constraint before 2412 release is used only for display sorting purposes. However, starting from 2412 release, if the sequence group property is non-empty on a given constraint, the sequence property determines the order or priority of the include constraint during the solve. If the sequence group property is empty, the sequence property continues to be used for display sorting as before.

Behavior of sequenced constraints in configuration

When include constraints are enabled, it supports the ordered execution of constraints using sequences and sequence groups. Sequenced include constraints are designed to ensure that within a group, at most one constraint can be entailed[3]. However, all constraints are considered together for satisfiability and entailment. A lower-weight constraint is enforced, meaning all respective higher-weight constraints are denied.

Sequence ordering applies only within a group of constraints that share the same sequence group property value.

Variability:

Family	Features
Models	**Luxury, Economy**
Fuel Type	**Electric, Petrol**
Engine	**X40, X60, X90**
Transmission	**Automatic, Manual**

3 Entailed is when the condition of a constraint is enforced either by user selection or by other constraints.

Include constraints with sequence and sequence group:

Constraint ID	Constraint type	Condition	Subject	Sequence	Sequence group
#1	**Include**	**Luxury and Electric**	**X40**	**10**	**Engine Group**
#2	**Include**	**Luxury**	**X60**	**20**	
#3	**Include**		**X90**	**30**	
#4	**Include**	**Luxury and X40**	**Automatic**	**10**	**Transmission Group**
#5	**Include**	**Luxury**	**Automatic**	**20**	
#6	**Include**		**Manual**	**30**	

Use case 1:

- User selection: **Luxury and Electric**
- Output: **Valid & Complete**
- Expanded expression: **Luxury and Electric** and **X40** and **Automatic**
- Explanation: The input **Luxury and Electric** triggers constraint **#1**, which has the highest weight in the **Engine Group** with a sequence of **10**, enforcing the **X40** feature. As a result, the combination of **Luxury and X40** triggers constraint **#4**, which has the highest weight in the **Transmission Group** with a sequence of **10**, enforcing the **Automatic** feature.

Use case 2:

- User selection: **Luxury**
- Output: **Valid & Complete**
- Expanded expression: **Luxury**
- Explanation: The input **Luxury** appears to satisfy constraint **#2**, but constraint **#1** remains unresolved and is not denied. The **Fuel Type** field still lacks a selection, allowing the user to choose **Electric** as the fuel type. Consequently, none of the constraints for the **Engine Group** are triggered, as a higher-weight constraint must be denied before a subsequent lower-weight constraint is considered. Similarly, no constraints are triggered in the **Transmission Group** either.

Use case 3:

- User selection: **Luxury** and **Petrol**

- Output: **Valid & Complete**
- Expanded expression: **Luxury and Petrol** and **X60 and Automatic**
- Explanation: The input **Luxury and Petrol** satisfies constraint **#2**, which has a lower weight compared to constraint **#1**. Constraint **#1** is denied in the **Engine Group** with sequence **20** and enforces the **X60** feature. Consequently, the combination of **Luxury and X60** satisfies constraint **#5**, which has a lower weight compared to constraint **#4**. Constraint **#4** is denied in the **Transmission Group** with sequence **20** and enforces the **Automatic** feature.

Use case 4:

- User selection: **Economy**
- Output: **Valid & Complete**
- Expanded expression: **Economy and X90 and Manual**
- Explanation: The input **Economy** satisfies constraint **#3**, which has a lower weight compared to constraints **#1** and **#2**. Constraints **#1** and **#2** are denied in the **Engine Group** with sequence **30** and enforce the **X90** feature. As a result, the combination of **Economy and X90** satisfies constraint **#6**, which has a lower weight compared to constraints **#4** and **#5**. Constraints **#4** and **#5** are denied in the **Transmission Group** with sequence **30** and enforce the **Manual** feature.

> Note:
>
> In modular configuration products, sequenced include constraints within the same sequence group that span across different modules are not supported in the 2412 release. In such cases, these constraints function as regular include constraints.

Copying configurator rules

You can copy configurator rules using the **Save As** command. The copied rule is associated to the source configurator context.

In Product Configurator, you can view the copied configurator rule along with the original rule and other previously created rules within the source configurator context.

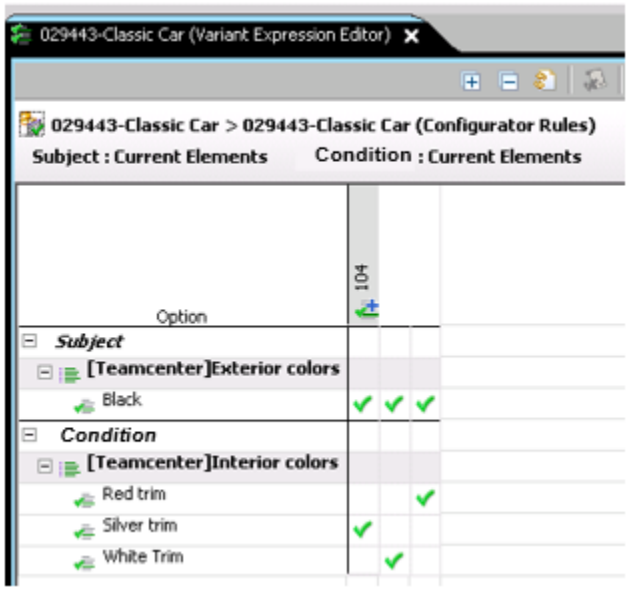
Splitting configurator rules

You can split a configurator rules expression in the **Variant Expression Editor** view. This allows you to specify multiple **Condition** criteria for the given **Subject** using a single rule. You can add multiple **Condition** columns for the same **Subject** to specify a variety of conditions when this rule become applicable.

Teamcenter splits the **Condition** section of the rule and creates another column along with the selected column and allows you to author a new expression. The newly created column is connected using a logical **OR** operator with the original expression column.

To split a rule, you right-click the rule column header and choose **Split Expression** .

For example, the include rule **104** states that if the interior color is red, silver or white, then you can only have a black car.



Once the rule is saved, you can only modify the **Condition** section for the split rules. The **Subject** columns appear grayed out because they are read-only with the exception of the first column with the original **Subject** expression. If you modify the **Subject** expression in the first column, it applies to all split expressions.

When the split rules are validated during save or on demand by clicking **Validate Variant Expressions** , Teamcenter generates a violation message only if *all* branches of the rule are invalid. The split rule is still considered valid even if one branch of the rule is valid.

> Note:
>
> Branches that display duplicate split rule information are removed during save.

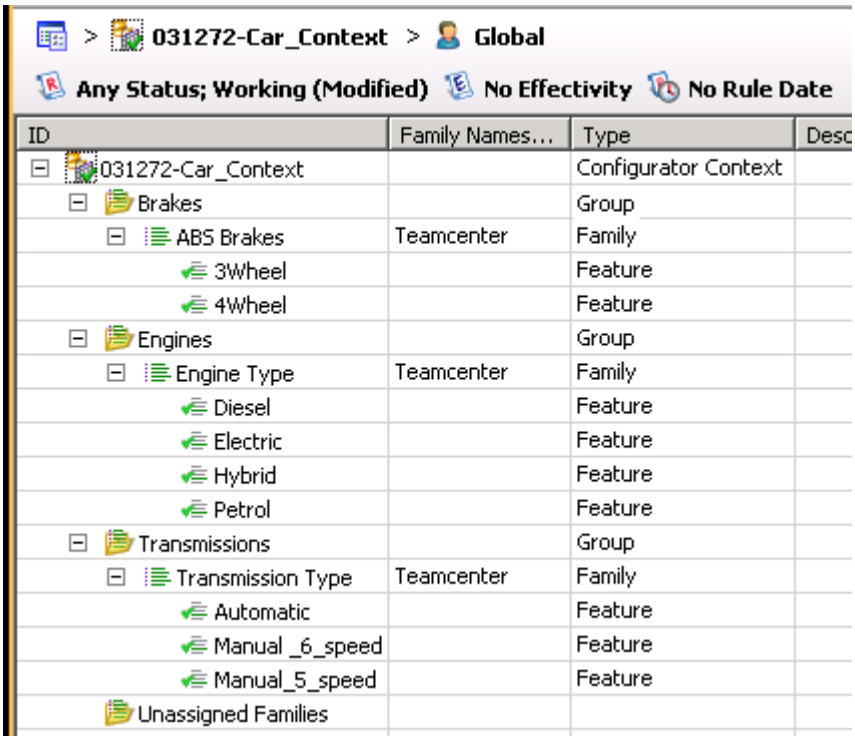
You can validate branches of the split rule as individual expressions by selecting the branches and choosing **Validate Individual Column Expression** . If the selected expression is invalid, the failed branch is displayed with in the header, and the validation error message appears. You can also view a tooltip with a violation message if you hover the cursor over the failed branch of the split rule.

Tip:

You can right-click the split rule column header and choose **Select Column Expressions** to select all expressions in the rule.

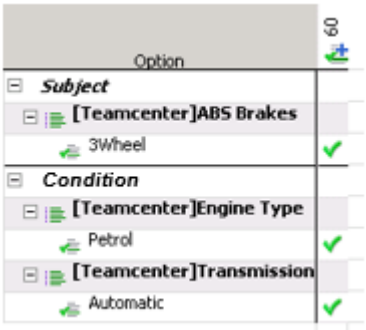
Example: Validating the configuration against an inclusion rule with and without a split

1. A positively biased configurator context is created with the following groups, families, and features.



ID	Family Names...	Type	Desc
031272-Car_Context		Configurator Context	
Brakes		Group	
ABS Brakes	Teamcenter	Family	
3Wheel		Feature	
4Wheel		Feature	
Engines		Group	
Engine Type	Teamcenter	Family	
Diesel		Feature	
Electric		Feature	
Hybrid		Feature	
Petrol		Feature	
Transmissions		Group	
Transmission Type	Teamcenter	Family	
Automatic		Feature	
Manual _6_speed		Feature	
Manual_5_speed		Feature	
Unassigned Families			

2. The following inclusion rule is added to allow **3Wheel** ABS brakes for cars with both the **Petrol** engine *and* the **Automatic** transmission.



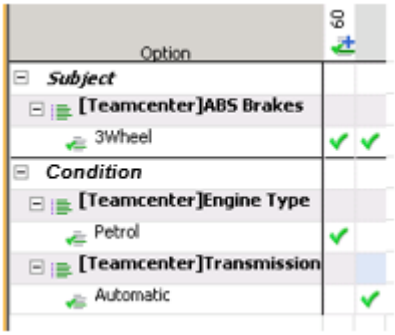
3. Open the **Variant Configuration** view and make your selections.

By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** mode.

The validation results are as follows.

Input	Validation result
Engine = Petrol & Transmission = Automatic & ABS = 3Wheel	Valid
Engine = Petrol & Transmission = Automatic & ABS = 4Wheel	Invalid
Engine = Diesel & Transmission = Automatic & ABS = 4Wheel	Valid
Engine = Petrol & Transmission = Manual_5_speed & ABS = 4Wheel	Valid

4. Next, split the inclusion rule to allow **3Wheel** ABS brakes for all cars with either the **Petrol** engine *or* the **Automatic** transmission.

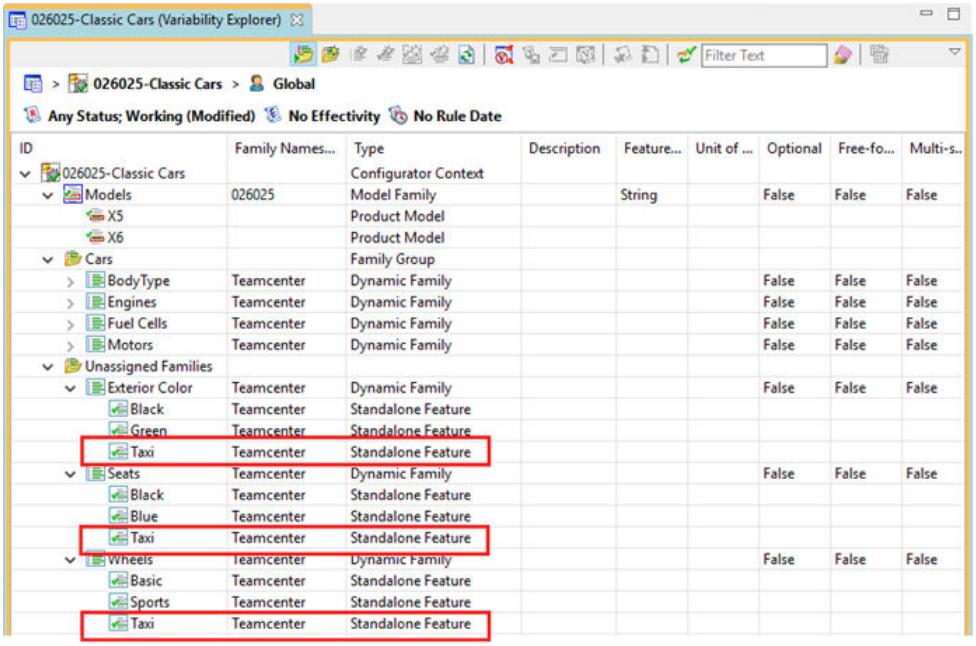


5. Open the **Variant Configuration** view to make your selections. The validation results are as follows:

Input	Validation result
Engine = Petrol & Transmission = Automatic & ABS = 3Wheel	Valid
Engine = Petrol & Transmission = Automatic & ABS = 4Wheel	Invalid
Engine = Diesel & Transmission = Automatic & ABS = 4Wheel	Invalid
Engine = Petrol & Transmission = Manual_5_speed & ABS = 4Wheel	Invalid

Example: Validating the configuration with standalone features

1. A positively biased configurator context is created with the following groups, families, and features. The **Classic Cars** configurator context contains dynamic families with standalone features. When configuring your content, you can specify a specific configuration that comes with seats, wheels, and color only available in a taxi. The **Taxi** standalone feature is available across three different families: **Exterior Color**, **Seats**, and **Wheels**.

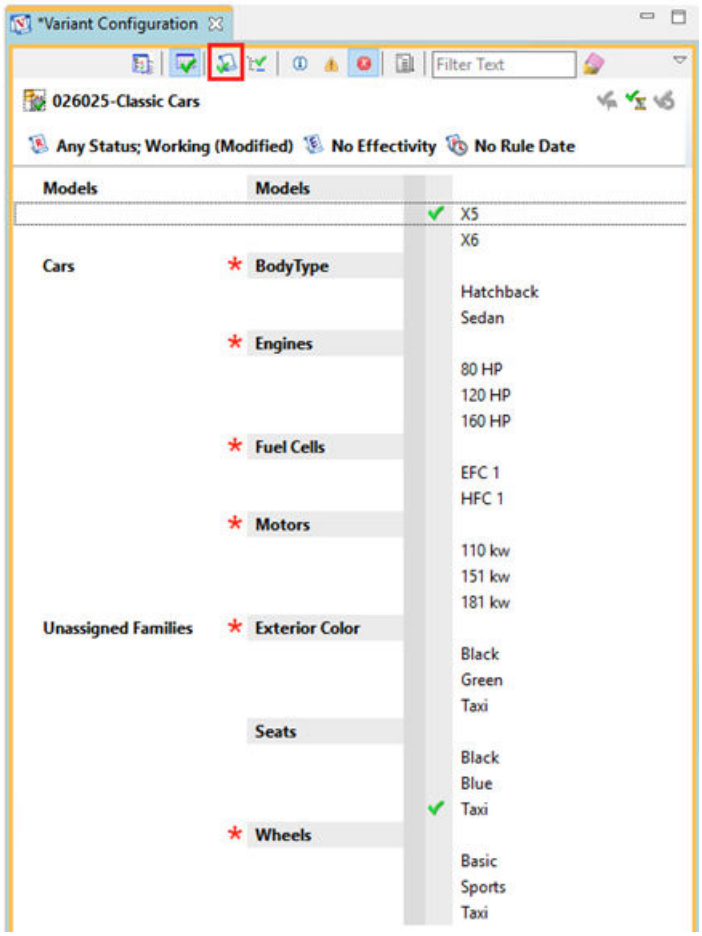


ID	Family Names...	Type	Description	Feature...	Unit of ...	Optional	Free-fo...	Multi-s...
026025-Classic Cars		Configurator Context						
Models	026025	Model Family		String		False	False	False
X5		Product Model						
X6		Product Model						
Cars		Family Group						
BodyType	Teamcenter	Dynamic Family				False	False	False
Engines	Teamcenter	Dynamic Family				False	False	False
Fuel Cells	Teamcenter	Dynamic Family				False	False	False
Motors	Teamcenter	Dynamic Family				False	False	False
Unassigned Families								
Exterior Color	Teamcenter	Dynamic Family				False	False	False
Black	Teamcenter	Standalone Feature						
Green	Teamcenter	Standalone Feature						
Taxi	Teamcenter	Standalone Feature						
Seats	Teamcenter	Dynamic Family				False	False	False
Black	Teamcenter	Standalone Feature						
Blue	Teamcenter	Standalone Feature						
Taxi	Teamcenter	Standalone Feature						
Wheels	Teamcenter	Dynamic Family				False	False	False
Basic	Teamcenter	Standalone Feature						
Sports	Teamcenter	Standalone Feature						
Taxi	Teamcenter	Standalone Feature						

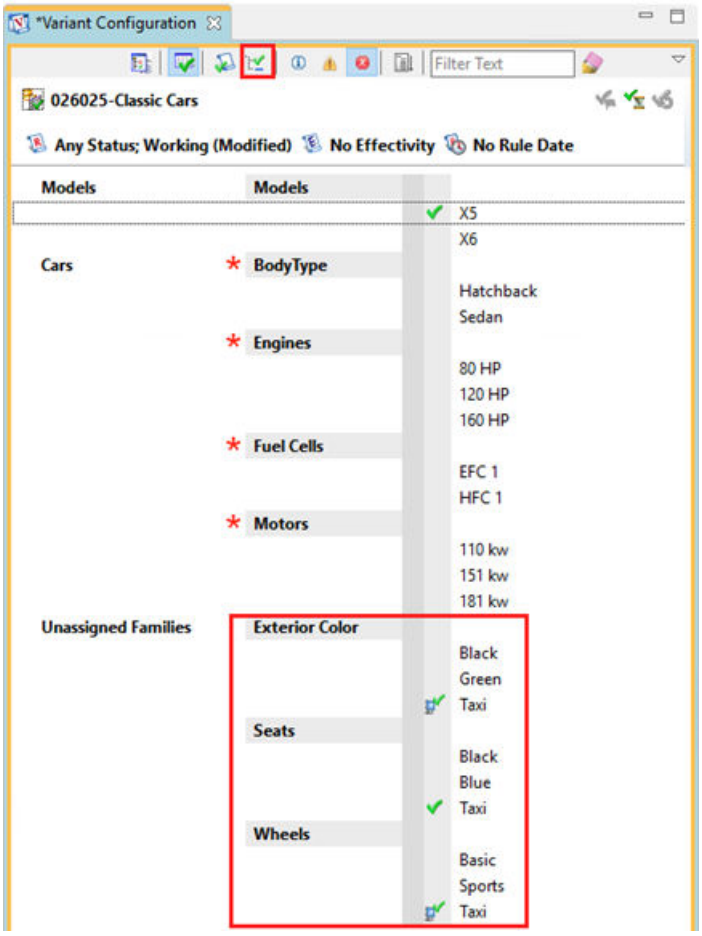
2. Open the **Variant Configuration** view and make your selections.

   By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** mode.

   Select the **X5** model and **Taxi** in the **Seats** family and click **Validate** to validate the configuration. Because it's a standalone feature, it is valid and implicitly selected across all the selections, including **Wheels** and **Exterior Color**.

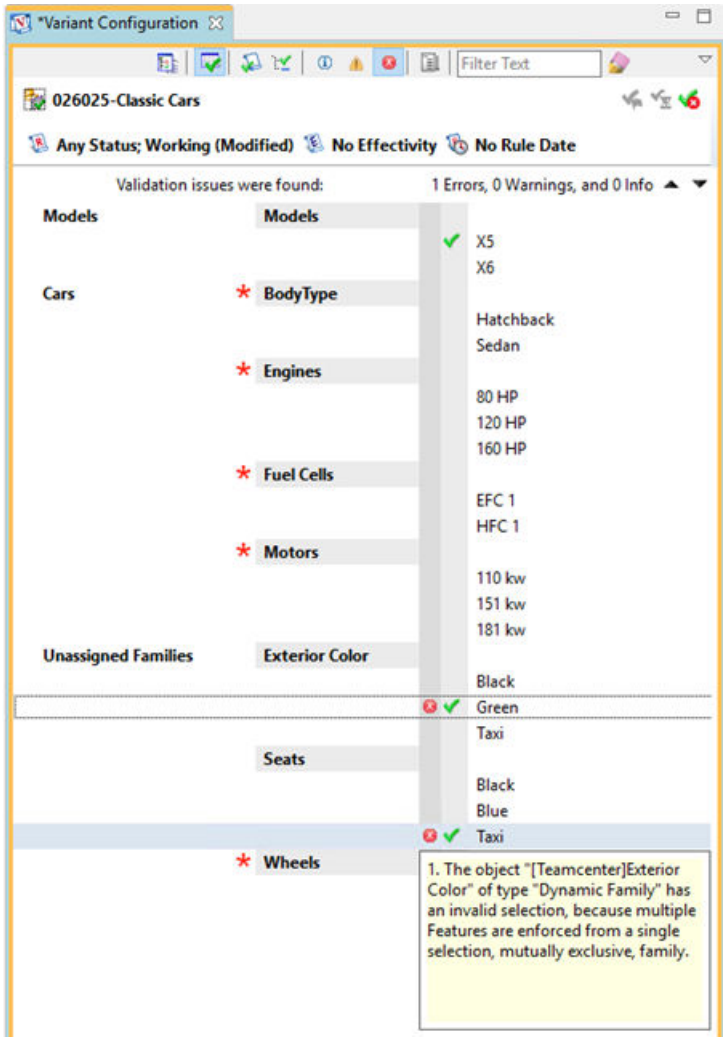


If you click **Expand**, the **Taxi** standalone features are displayed as system selections in the **Wheels** and **Exterior Color** dynamic families.



3. Next, clear your configuration expression. Select the **X5** model and **Taxi** in the **Seats** family. Additionally, select a specific car exterior color, not the one that comes with a taxi, such as **Green** from the **Exterior Color** family.

   When you click **Validate**, the system generates an error because **Taxi** is already selected in the dynamic families. You cannot make more than one selection in the dynamic family. By default, features that are organized in the dynamic family are mutually exclusive.



4. To change the mutually exclusive behavior for **Taxi** in the **Seats** family, you create an exception rule that grants an exception for **Taxi** in the **Seats** family.

ID	Type	Severity	Message	Subject	Applicability	Is Global	E
310	Exception Rule	Error	X5-> Taxi is an exception in Seat	[Teamcenter]Seats = Taxi	[026025]Models = X5	False	

5. Open the **Variant Configuration** view and make your selections.

   By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** mode.

   The validation results are as follows.

Input	Validation result
Model = X5 & Exterior Color = Taxi & Seats = Blue	Valid
Model = X5 & (Seats = Blue OR Seats = Taxi)	Valid
Model = X5 & Exterior Color = Green & Seats = Taxi	Invalid

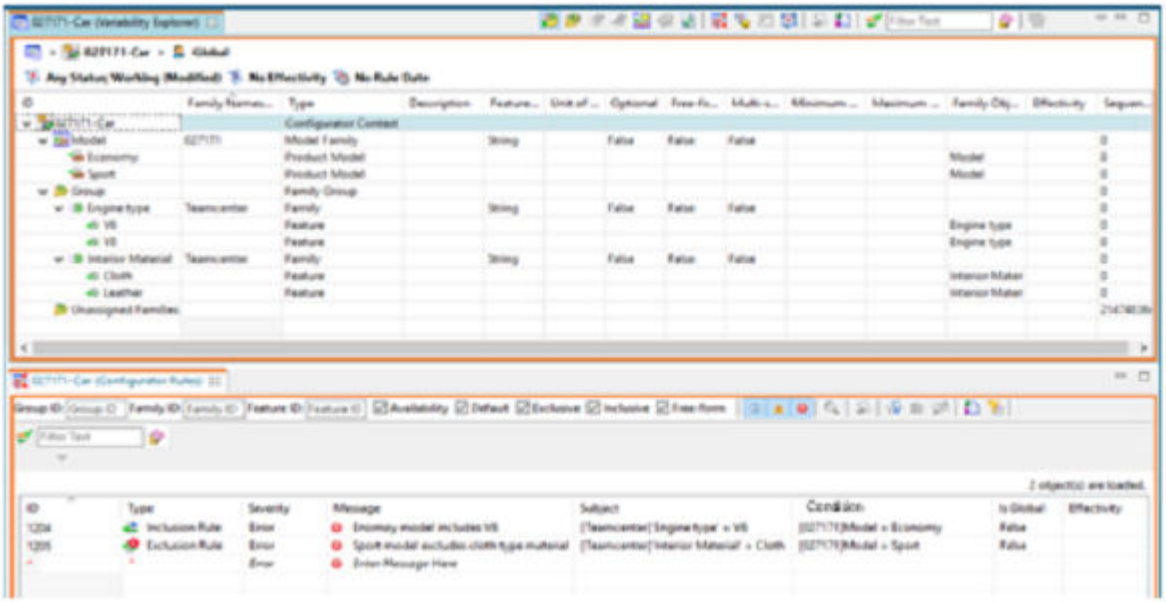
Example: Apply constraints and allow validation rules to expand

The **Apply Constraints** selection in the **Variant Configuration** view allows users to select if all the constraints should be applied along with user selections while applying the configuration on content. If it is deselected, only user selections are considered while applying the configuration from **Variant Configuration** view on content.

The **Allow Validation Rules to Expand** selection in the **Variant Configuration** view allows users to select if validation rules should expand and perform selection during the expansion process. If only one selection is possible, these rules (exclusion rule and availability rule) can expand and perform selection. If deselected, expand does not make selections based on validation rules.

By default, Teamcenter displays the **Variant Configuration** view with **Apply Constraints** and **Allow Validation Rules to Expand** as selected. If users deselect any of the two buttons, closes the view, and relaunches the view, these buttons are reset to the default state.

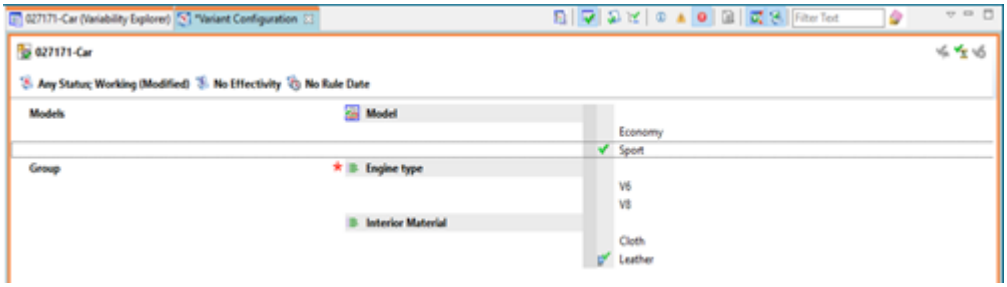
1. A positively biased configurator context is created with the following data.



2. Open the **Variant Configuration** view and make selections as **Model=Sport** with the **Allow validation rules to expand** option as *selected*.

3. To perform the validate and expand action, click **Validate** and then click **Expand**.

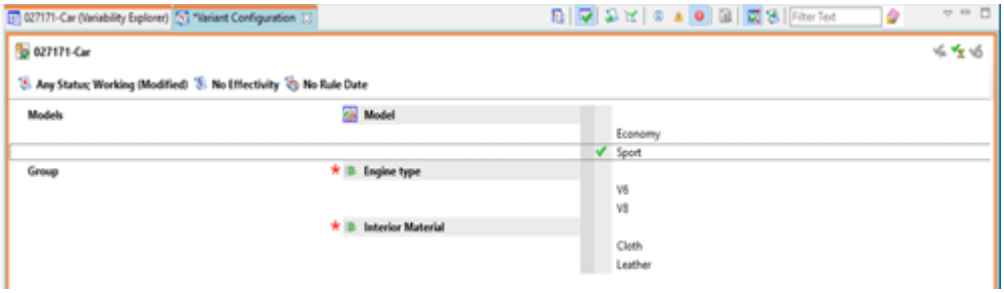
   The expansion action expands to system selection of **Interior Material=Leather** as **Cloth** is excluded due to the exclusion rule and only one possible selection is allowed other than what is available for the **Interior Material** family.



4. Open the **Variant Configuration** view and make selections as **Model=Sport** with the **Allow validation rules to expand** option as *deselected*.

5. To perform the validate and expand action, click **Validate** and then click **Expand**.

   The expansion action does not expand to the system selection of **Interior Material=Leather**.



Checking variant constraints

Variant constraints (including derived feature rules and compatibility constraints) form a complex network of logic that may potentially include circular references, rules that can never be satisfied, and rules that never result in a selected feature for a valid set of families. Teamcenter detects these conditions when they exist, allowing you to ensure that your rules result in complete and consistent results for all combinations.

For example, Teamcenter provides feedback if any of the following situations exist:

- A rule (independent of any other rules) can never be satisfied.
- A rule (independent of any other rules) is always true.
- For a product model, a set of rules when executed together can never be satisfied.
- For a product model, a set of rules when executed together result in an endless chain of logic — a circular reference.

Teamcenter provides feedback as you work if your current selections would not provide any valid combination of selections for the remaining mandatory families. It also provides feedback on the

remaining number of valid possible configurations based on the selections made so far in the configuration process.

Authoring free-form rules

About free-form rules

In Product Configurator, you can create configurator rules, such as **default, exclusion, exception, inclusion, and availability rules**. These rules allow you to create a variant expression that sets one or more features, if any stated preconditions are satisfied, and build a relationship between the **Subject** and the **Condition** expressions. All interactions with the standard configurator rules are grid-based and table-based.

In addition to configurator rules, you can create more complex free-form SMT-based rules using the SMT Lib scripting language. They do not contain the **Subject** and the **Condition** expressions. Instead, they allow you to build a complex expression. A free-form rule is created and modified in a multiline editor. Similar to standard configurator rules, free-form rules can be revised and can have effectivity and severity. Using free-form rules, you can bundle multiple constraints under one single revisable WSO with one owner, one approval process, one responsibility, and one effectivity when it has the same severity level. In contrast to standard rules, free-form rules are not validated in the **Configurator Rules View**.

> Note:
>
> The ability to create free-form rules is disabled by default. Your administrator must use the **Cfg0EnableFreeFormRuleSupport** preference to enable this functionality. By default, it is set to **true**.

Free-form rules allow you to:

- Establish family to family and feature to feature constraints

  > Example:
  >
  > `“Always: [CAR]Weight > [TRAILER]Weight” •“If [CAR]TowingPackage = true: [CAR]Weight > [TRAILER]Weight”`

- Execute calculations

  > Example:
  >
  > `“[CAR]Weight > [TRAILER]Weight + [BOAT]Weight”`

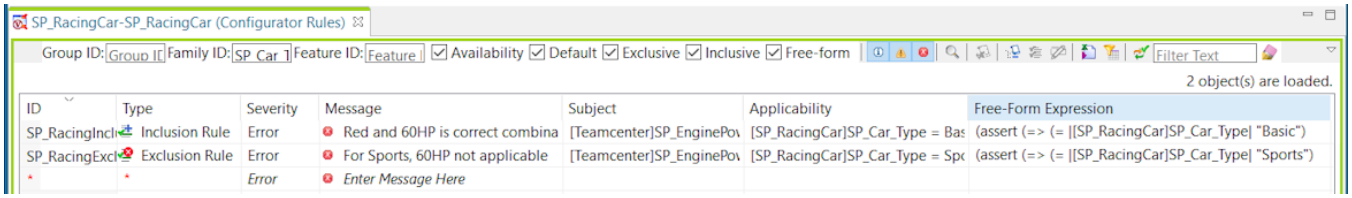
- Build XOR rules

Example:

```
(assert ( ite a (!b) (ite b (!a) a)) )
```

- Create equivalence rules

In addition to standard columns in the **Configurator Rules** view, you use the following column:



- **Free-Form Expression**

  It contains an expression as per industry standard language for a constraint solver. You can either click **Open Free-form Expression Editor** or click the corresponding cell under the **Free-Form Expression** column to open the **Free-form Expression Editor** dialog box that allows you to create or modify free-form rules. The product configuration is evaluated against all such configurator constraints applicable to the product to determine if it is a valid or invalid configuration. It updates the configuration, if necessary, to make it valid.

  For example, the `(assert ( ite a (!b) (ite b (!a) a)) )` statement below represents an XOR relation between independent features.

  Note:

  `“assert ( => ( > ( * length width) 100) ( = TruckModel M1))” to represent business constraint where (length * width) > 100 then include TruckModel = M1.`

  By default, the system generates the read-only SMT expression in the **Free-Form Expression** column for standard exclusion and inclusion rules created using the grid format. However, the package, exception, and availability rules do not display it. You can modify the SMT expression for free-form rules only.

Free-form rules can only be modelled like Inclusion or Exclusion type

Free-form rules cannot be modelled like **Availability**, **Default**, and **Exception** types. Rules of these types require separation between **Subject** and **Condition**.

Free-form rules do not have distinction between **Subject** and **Condition**. Together, they form one assert statement.

Example:

The Availability Rule of the same **Subject** together form a concrete statement.

A1 is Available to M1.

A1 is Available to M2.

This means that A1 is Available to "M1 | M2". However, for every other combination, A1 is not available.

Similarly, there is a different logic with **Default** and **Exception** rules that require the explicit distinction of **Subject**. Therefore, free-form rules can only be modelled like **Inclusion** or **Exclusion** type.

Best practices for authoring free-form rules using the SMT standard expression

Siemens Digital Industries Software recommends observing the following best practices when using the SMT Lib scripting language to create free-form rules.

- Global free-form rules do not support multi-select families in SMT expressions.
- You can only use the `assert` statement. Other statements are not supported. In addition, you cannot use keywords as variable names, such as `declare` and `push`.
- When building your SMT expressions, use the following syntax to identify the family variable.

  ```
  |[Namespace]familyA|
  ```

  For example, `|[Teamcenter]Interior_color|`.

  Do not use the vertical bar | or quotation marks “” in any of the family or feature names if you plan to use those in the free-form SMT expression.

- For a multi-select family, use the following syntax to select one of its feature values.

  ```
  |[Namespace]familyA|value| = true
  ```

  For example, `|[[Teamcenter]Mirror]Left| = true`.

  - `Teamcenter` is the namespace.
  - `Mirror` is the variable family.
  - A feature value selected from that multi-select family is `Left`.

- Using string families in free-form rules will have an impact on performance.

- For float type families, use the new SMT2Lib macros provided by Siemens Digital Industries Software. These macros are to be used for checking the equality of float type expressions.

SMT2Lib Expression	Meaning
`(EQ_eq Family Value)`	; Return SMT2Lib expression for Family Expr is equal to Value Expr
`(EQ_ne Family Value)`	; Return SMT2Lib expression for Family Expr is not equal to Value Expr
`(EQ_lt Family Value)`	; Return SMT2Lib expression for Family Expr is less than to Value Expr
`(EQ_le Family Value)`	; Return SMT2Lib expression for Family Expr is less than or equal to Value Expr
`(EQ_gt Family Value)`	; Return SMT2Lib expression for Family Expr is greater than Value Expr
`(EQ_ge Family Value)`	; Return SMT2Lib expression for Family Expr is greater than or equal to Value Expr

  This syntax is recommended to ensure consistency of results.

- The length of a string family cannot exceed 128 characters. Therefore, the result of an expression of any string operation such as string concatenation cannot exceed 128 characters.

- When using the date type family in your expressions, use the *Epoch format*.

- Use of quantifiers negatively impacts system performance.

  Siemens Digital Industries Software recommends using quantifier-free formulas over linear integer and real arithmetic ("QF_LIRA"). The list of supported logic formulas may change in a future version of Teamcenter.

- Uninterpreted functions are not allowed.

- A good resource for additional information about the SMT-LIB input/output language can be found at **SMT-LIB website**.

The SMT syntax examples of various types of free-form rules

The following free-form rule examples illustrate the SMT Lib scripting syntax for various family types.

Family type or operation	Syntax	Example
Multi-select family	`|[[Namespace]family A]feature| = true`	`|[[Teamcenter]Mirror]Left| = true.`

Family type or operation	Syntax	Example
		• `Teamcenter` is the namespace. • `Mirror` is the variable family. • A feature value selected from the multi-select family is `Left`.
Regular families	`(assert(= |[Namespace|(* |[Namespace]Family A||[Namespace]Family B| )))`	`(assert(= |[SMTLib]Area| (* |[SMTLib]Length for area calculation||[SMTLib]Width for area calculation| )))` `(assert (=> (= |[SMTLib]FuelType|"Electric")(= |[SMTLib]Fuel Tank Capacity| 1000)))`

Family selection	• **Integer Family**
	`(assert (=> (= |[Namespace]Family A|"Feature A") (= |[Namespace]FamilyB|0)))`
	• **String family with the selection of None**
	`(assert (=> (= |[Namespace]Family A|"Feature A") (= |[Namespace]Family C|"")))`
	• **String family with the selection of Any**
	`(assert (=> (= |[Namespace]Family A| "Feature A") (distinct |`
		• **Integer Family**
		`(assert (=> (= |[SMTLib]FuelType|"Electric") (= |[SMTLib]Fuel Tank Capacity| 0)))`
		• **String family with the selection of None**
		`(assert (=> (= |[SMTLib]FuelType|"Electric") (= |[SMTLib]Fuel Tank String| "")))`
		• **String family with the selection of Any**
		`(assert (=> (= |[SMTLib]FuelType|"Electric") (distinct |[SMTLib]Fuel Tank String| "")))`

Family type or operation	Syntax	Example
	`[Namespace]Family C| ""))))`	
Standalone feature	(assert(=> (= |[Configurator context ID]Model| "Model Name") (= |[Namespace]Std feature A| true)))	`(assert(=> (= |[SMT-100057]Model| "Ertiga-C01") (= | [SMTLib]Taxi| true)))`
String operation (concatenation)	`(assert (=> (= | [Configurator context ID]Model| "Model Name") (= |[Namespace]Family D| (str.++ "Model:" " Model Name" )))))`	`(assert (=> (= |[SMT-100057]Model| "Ertiga-C01") (= | [SMTLib]Order| (str.++ "Model:" " Ertiga" )))))`

Package Family/Summary family (similar to a regular family authoring)	(assert(=> (= |[Configurator context ID]Model| "Model Name") (= | [Namespace]Package A| "Feature")))
	If the package family is multiselect:
	(assert(=> (= |[Configurator context ID]Model| "Model Name") (= |[[Namespace]Package A]Feature| true)))
		`(assert(=> (= |[SMT-100057]Model| "Ertiga-C01") (= |[SMTLib]Safety| "Standard")))`
		If the package family is multiselect:
		`(assert(=> (= | [SMT-100057]Model| "Ertiga-C01") (= | [[SMTLib]Safety]Standard| true)))`

Dynamic family level selection	• **Equal =** (= |[Namespace]Dynamic Family| 0) • **Not equal !=** (!= |[Namespace]Dynamic Family| 0)	• **Equal =** (= |[Teamcenter]Color| 0) • **Not equal !=** (!= |[Teamcenter]Color| 0)
Date type family	For date families, specify the date in the Epoch format.	The date 3/13/2020 is converted to 1584089120 in the Epoch format. (assert(=> (= |[SMT-100057]Car Model Family| "Ertiga-C01") (= |[SMTLib]DateFamily| 1584089120)))
Float type family	For float families, use the SMT2Lib macros provided.	`(assert(=> (= |[000042]Models| "M1" ) ( EQ_eq`

Family type or operation	Syntax	Example
	`(assert(=> (= |[Namespace]Model| "Model Name") (EQ_eq |[Namespace]FloatFam| value)))`	`|[000042]Length| 10.8 )))`

SMT rule authoring examples

SMT and strings

- Any expression that is more than a single term should be surrounded by brackets

  (assert(= LeftExpression RightExpression**))**

- Feature-Value Allocated to Feature-Family is read using:

  |[Namespace]FamilyName|

- Literal string values should be surrounded by double quotes.

  (assert(= |[Namespace]StringFamilyName| "LiteralStringValue"))

- No brackets around function argument list.

  (FunctionName Expression1 Expression2 Expression3 ...)

List of known SMT string functions:

- **str.++** String concatenation
- **str.len** String length
- **str.indexof** Index of characters within strings (starts with 0)
- **str.contains** Does string contain substring
- **str.prefixof** Does string start with substring
- **str.suffixof** Does string end with substring
- **str.at** Character at the given index of a string
- **str.to.int** Convert string to integer (atoi)

Sample configurator context:

Family Name	Type	Values
FamA	String	**"A1 , A2"**
FamB	String	**"B1 , B2"**
FamAB	String	Free form
FamC	String	Free form
Color	String	**Red,Green,Blue**
FamE	Int	Free form
FamF	Int	Free form
FamD	Int	Free form
isGreen	Boolean	Family
isRed	Boolean	Family

Sample free form rules:

- **(assert(= |[000137]FamAB| (str.++ |[000137]FamA| |[000137]FamB|)))**

  FamAB = String Concatenation of FamA and FamB

- **(assert(= |[000137]FamC| "C"))**

  FamC = "C"

- **(assert(= |[000137]FamD| (str.len |[000137]Color|)))**

  FamD = String Length of Color selected

- **(assert(= |[000137]FamE| (str.indexof |[000137]Color| "e" 1)))**

  FamE = Index of first instance of character e in Color selected

- **(assert(= |[000137]isGreen| (str.contains |[000137]Color| "ee")))**

  isGreen = Does Color contain string "ee"

- **(assert(= |[000137]isRed| (str.prefixof "r" |[000137]Color|)))**

  isRed = Does Color start with character "r"

- **(assert(= |[000137]isRed| (str.suffixof "d" |[000137]Color|)))**

isRed = Does Color end with character "d"

- **(assert(= |[000137]FamF| (str.to.int (str.at |[000137]FamA| 1))))**

  famF = Convert String to Integer of character at index 1 of FamA

Simple SMT constructs

- **(assert (=>** Condition Subject **))**
- Condition and subject is written in the following format:

  (= Family LiteralValue)
- Float families are an exception where the **EQ_eq** function is used:

  (EQ_eq FloatFamily FloatLiteralValue)
- For multiselect family to ensure selection of feature, **=** operator is not needed:

  |[[NS]Family**]**Feature**|**

  For ensuring no selection of feature apply not operator:

  (not |[[NS]Family**]**Feature**|)**
- For **Exclusion** rule, add the not function to subject expression:

  (**not** IncludeExpression)
- For standalone features:

  (assert (= |[NS]StandaloneFeatureName**| true))**

Examples:

- **[NS]Models = M1 Includes [NS]FloatFam = 0.1**

  (assert (=> (= |[NS]Models| "M1") (EQ_eq |[NS]FloatFam| 0.1)))
- **[NS]IntegerFam = 10 Includes [NS]StringFam = "Ten"**

  (assert (=> (= |[NS]IntegerFam| 10) (= |[NS]StringFam| "Ten")))
- **[NS]FloatFam=3.14 Includes [NS]BoolFam = True**

(assert (=> (= |[NS]FloatFam| 3.14) (= |[NS]BoolFam| true)))

- **[NS]BoolFam = False Excludes [NS]FloatFam=3.14**

  (assert (=> (= |[NS]BoolFam| true) (not (= |[NS]FloatFam| 3.14))))

- **[NS]M1 Includes [NS]Taxi**

  (assert(=> (= |[NS]Models| "M1") (= |[NS]Taxi| true)))

Define free-form configurator rules

In addition to **configurator rules**, you can create complex free-form SMT-based rules using the SMT Lib scripting language. Using free-form rules, you can express restrictions and constraints that are not limited to a Teamcenter expression format for a standard configurator rule.

The ability to create free-form rules is disabled by default. Your administrator must use the **Cfg0EnableFreeFormRuleSupport** preference to enable this functionality. By default, it is set to **True**.

1. Open the configurator context or the dictionary in the **Variability Explorer** view, and then click **Open Configurator Rules view** to open the configurator rules editor.

   Teamcenter displays the selected editor.

2. Click in the **ID** field of the empty row and type the ID of the rule, or leave it blank so that the next available ID is automatically assigned.

3. Click the **Type** field in the same row.

   Teamcenter displays a list of all configurator rule types.

   Your administrator can create company-specific subtypes of configurator rules in Business Modeler IDE.

4. Select the **Free-form Rule** type.

5. Click the **Severity** field in the same row.

   Teamcenter displays a list of all severity types.

6. Press the **Enter** key or click anywhere in the view.

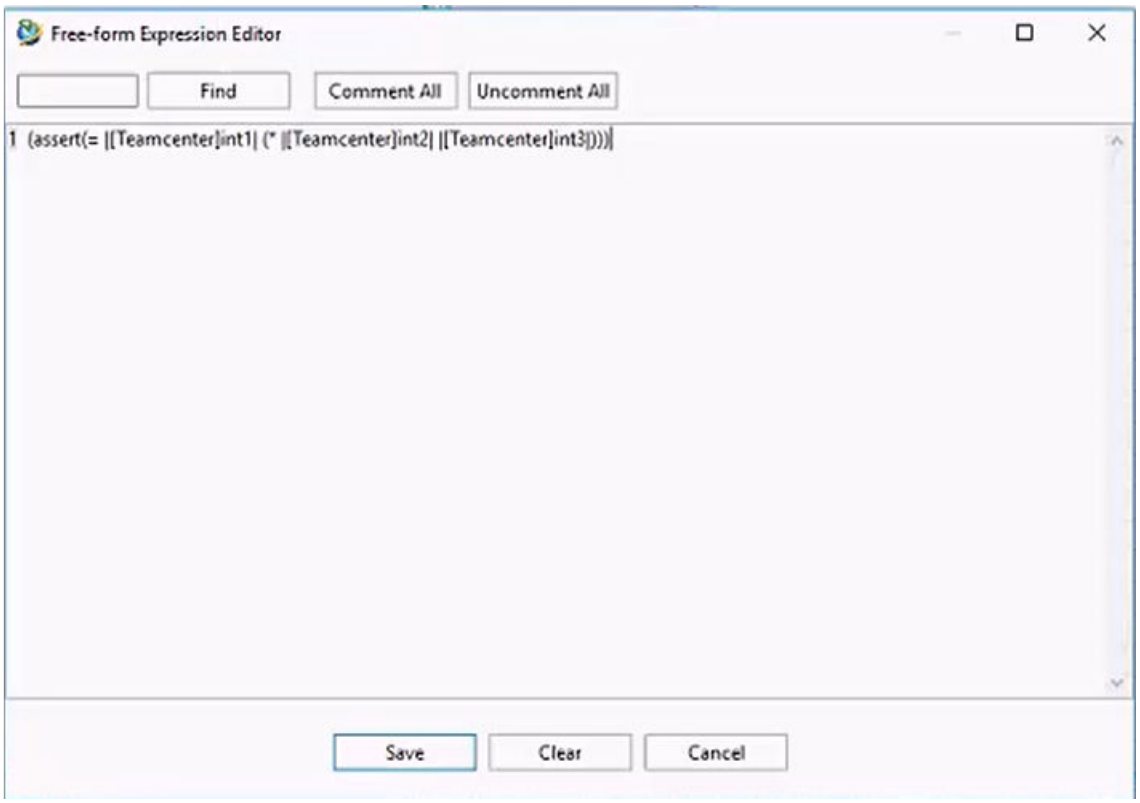
7. Click **Save** on the main toolbar to save this newly created rule.

8. You can either click **Open Free-form Expression Editor** or click the corresponding cell under the **Free-Form Expression** column to open the **Free-form Expression Editor** dialog box that allows you to create or modify free-form rules.

   For free-form rules, the command to display **Variant Expression Editor** is disabled.

   By default, the **Free-Form Expression** column is not visible. You can add columns by right-clicking the menu bar, choosing **Column**, and then modifying the list of displayed columns in the **Column Management** dialog box.

9. Define the complex rule expression using the industry standard language for a constraint solver. If the expression is incorrect, you cannot save and close the editor.



   The following is an example of a rule expression. The example in the screenshot and the example below defines the exact same constraint.

```
1       (assert
2           (let
3               (
4                   (TheaterCapacity |[Teamcenter]int1|)
5                   (RowCount |[Teamcenter]int2|)
6                   (SeatCount |[Teamcenter]int3|)
7               )
8               (= TheaterCapacity ( RowCount * SeatCount )
9           )
10      )
```

Click **Comment All** to comment out selected lines. The system ignores comments during processing. The comments are displayed in green. To include them, select one or many lines and click **Uncomment All**.

The system also displays the equivalent expression in the SMT format for standard configurator rules. However, you can only view but not modify the expression. The default and availability rules do not display the SMT expression.

10. Click **Save** on the main toolbar to save changes.

11. For the standard inclusion and exclusion rules only, you can view the SMT expression in the **Free-Form Expression** column. By default, the system converts inclusion and exclusion rules into a read-only free-form expression.

    You can copy a free-form expression generated from the include and exclude rule and use it as a base for the new free-form rule. Once that expression is used as a free-form rule, you can modify it as needed. If you make changes in the include or exclude rule free-form expression, the system does not update the **Subject** and **Condition** expressions accordingly. However, if you change **Subject** and **Condition**, the system updates the expression.

    To convert the include and exclude rule into a free-form rule.

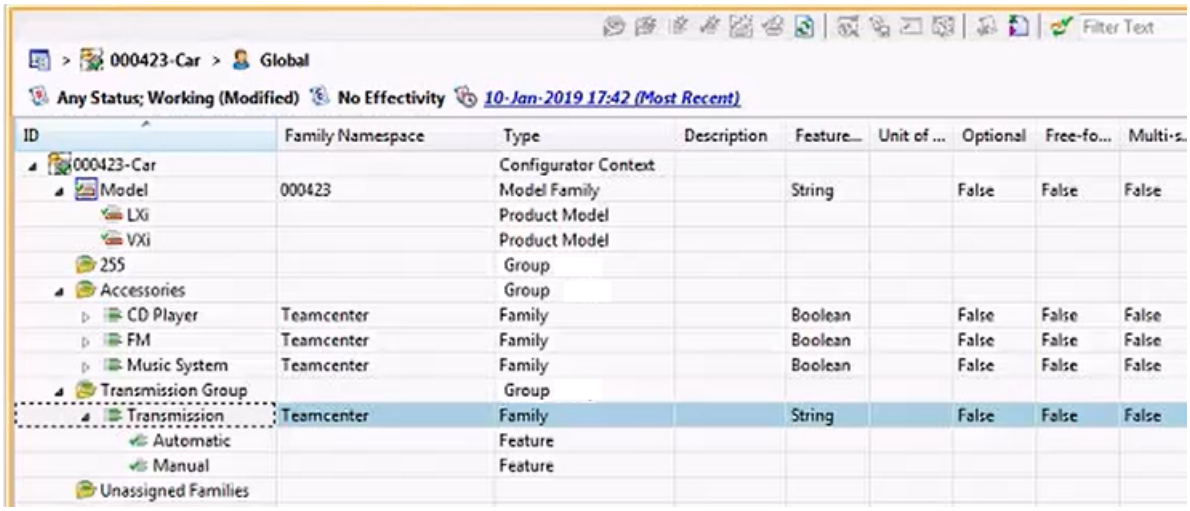
    - Locate the standard include or exclude rule from which you want to create a more complex free-form rule.
    - Click to open the **Free-form Expression Editor**.
    - Select and copy the expression generated by the system.
    - Create a brand new free-form rule and paste that expression into the **Free-form Expression** field.
    - Modify the rule as required.

- Delete the original standard include or exclude rule, if needed.

> Tip:
>
> You may need to reset the **Rule Date** to today, clear the date or use the required date in order to view your availability rules. You work on a snapshot or a frozen set of the configurator data and use that data to get consistent results for a period of time. These results are not impacted by subsequent changes, such as additions or updates of features and rules.

Example: Using a free-form rule with effectivity to establish feature to feature constraints

1. A positively biased configurator context is created with the following groups, families, and features.

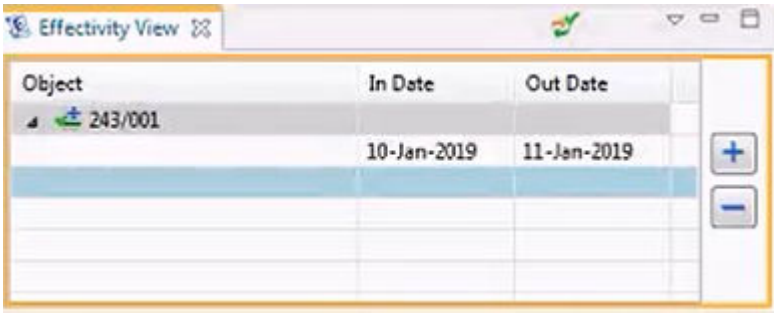


ID	Family Namespace	Type	Description	Feature...	Unit of ...	Optional	Free-fo...	Multi-s...
000423-Car		Configurator Context						
Model	000423	Model Family		String		False	False	False
LXi		Product Model						
VXi		Product Model						
255		Group						
Accessories		Group						
CD Player	Teamcenter	Family		Boolean		False	False	False
FM	Teamcenter	Family		Boolean		False	False	False
Music System	Teamcenter	Family		Boolean		False	False	False
Transmission Group		Group						
Transmission	Teamcenter	Family		String		False	False	False
Automatic		Feature						
Manual		Feature						
Unassigned Families								

2. The following free-form rule is created, stating that **LXI** models include **Automatic** transmission.

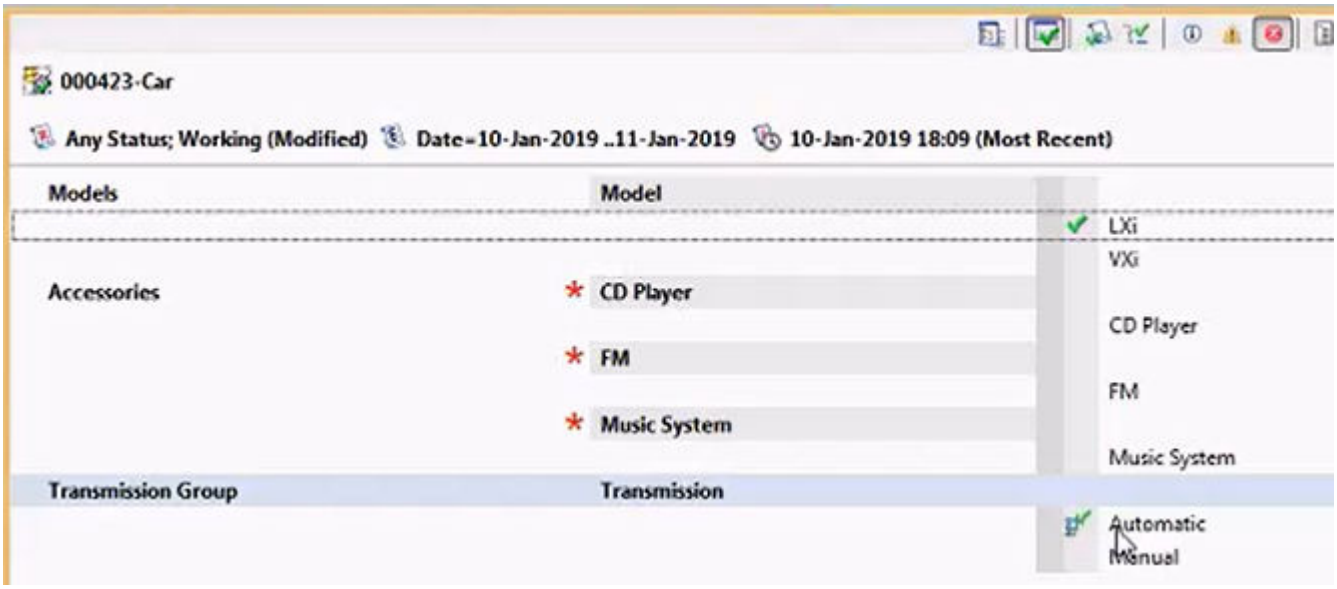
```
(assert( => (=[000423]LXI) (= [Teamcenter]Automatic)))
```

3. Next, add the following effectivity to your rule.



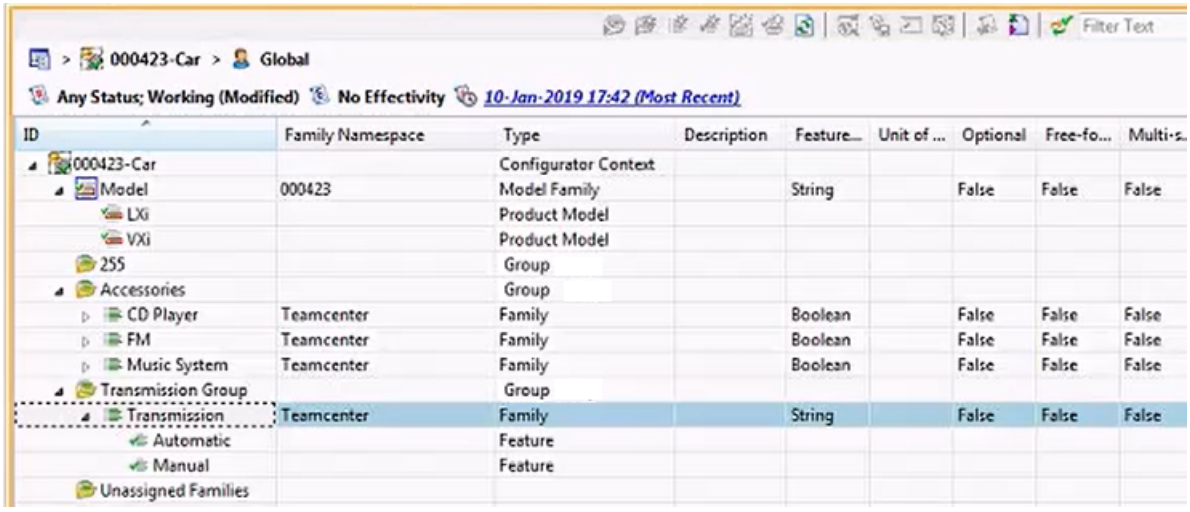
4. Open the **Variant Configuration** view to make your selections. Select **LXI**, set the effectivity to January 10, and click **Expand**.

The system selects **Automatic** in the **Transmission** family based on the defined free-form rule.



Example: Using a free-form XOR rule

1. A positively biased configurator context is created with the following groups, families, and features.



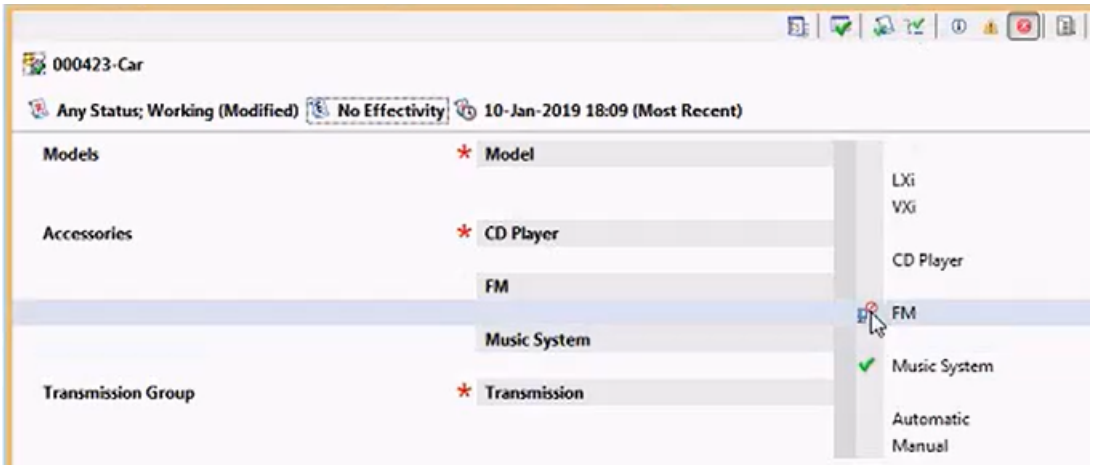
ID	Family Namespace	Type	Description	Feature...	Unit of ...	Optional	Free-fo...	Multi-s...
000423-Car		Configurator Context						
Model	000423	Model Family		String		False	False	False
LXi		Product Model						
VXi		Product Model						
255		Group						
Accessories		Group						
CD Player	Teamcenter	Family		Boolean		False	False	False
FM	Teamcenter	Family		Boolean		False	False	False
Music System	Teamcenter	Family		Boolean		False	False	False
Transmission Group		Group						
Transmission	Teamcenter	Family		String		False	False	False
Automatic		Feature						
Manual		Feature						
Unassigned Families								

2. The following XOR rule deselects FM if a music system is selected and vice versa. The use of this free-form rule eliminates the need to build a complex standard rule with the `If -> Then` statement.

```
(assert(ite [Teamcenter]Music System) (not [Teamcenter]FM)
[Teamcenter]FM))
```

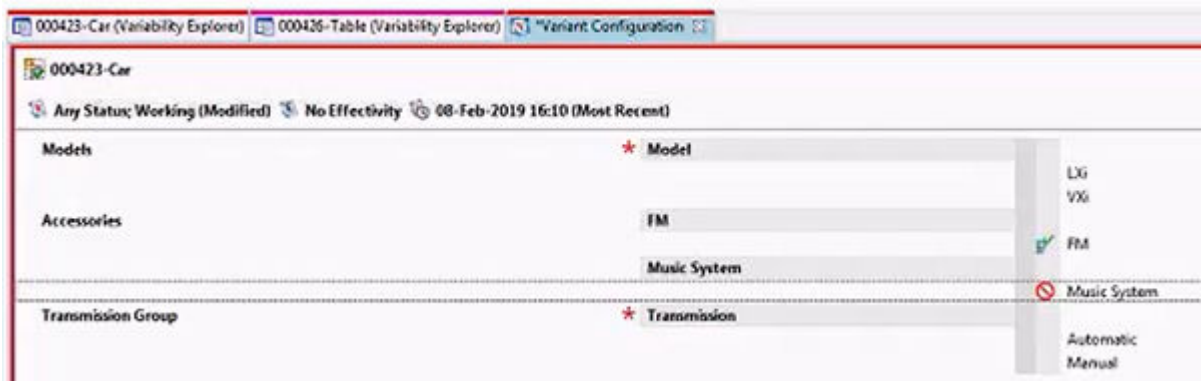
3. Open the **Variant Configuration** view to make your selections. Select **Music System** and click **Expand**.

The system deselects FM in the FM family based on the defined free-form rule.



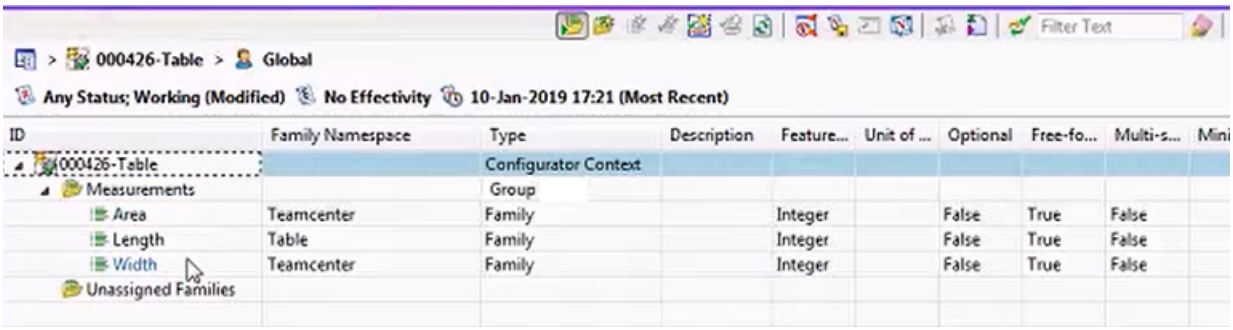
4. Select **FM** and click **Expand**.

The system deselects **Music System** based on the defined free-from rule.



Example: Using a free-form calculation rule

1. A positively biased configurator context is created with the following group with a free-form family and integer features.

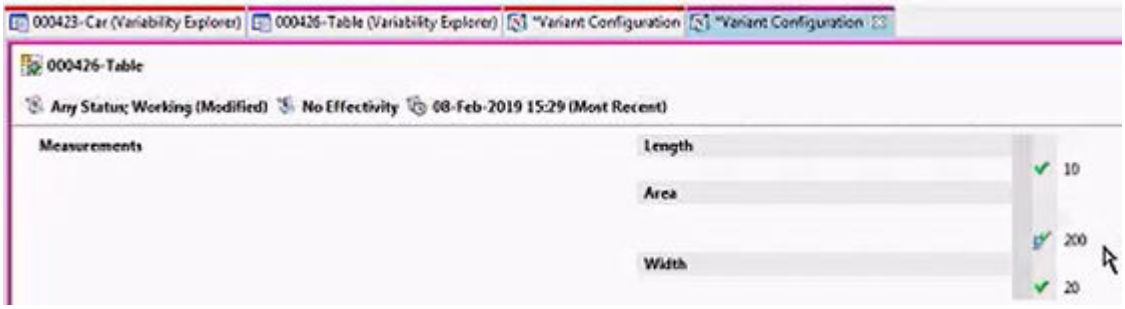


2. The following free-form calculation rule is created to calculate area, length, or width, based on the values presented in the formula.

```
(assert(= [Teamcenter]Area)(*[Table]Length[Teamcenter]Width)))
```

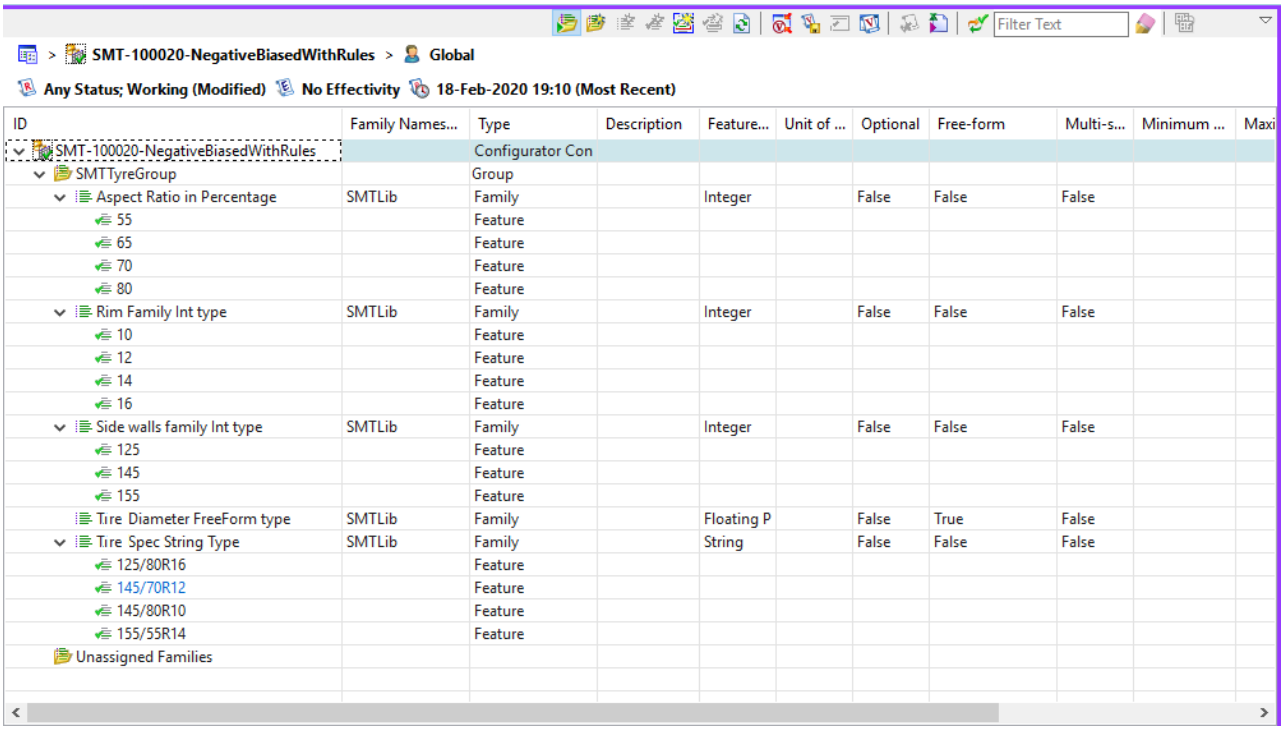
3. Open the **Variant Configuration** view to set your values. Set **Length** to **10** and **Width** to **20**. Click **Expand**.

   The system calculates the area. You can set any of the integer values, such as area, length, or width, and the system calculates the resulting value based on the defined free-form rule because this rule is presented as a formula.



Example: Using a free-form rule

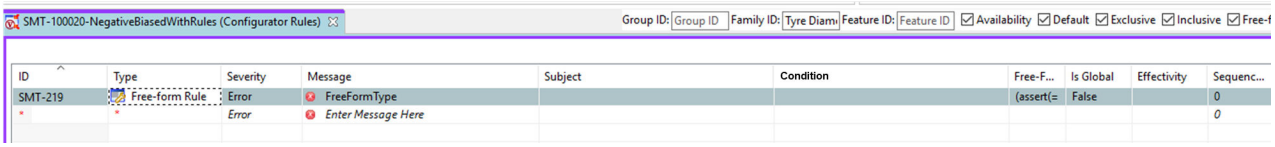
1. A negatively biased configurator context is created with the following groups, families, and features.



ID	Family Names...	Type	Description	Feature...	Unit of ...	Optional	Free-form	Multi-s...	Minimum ...	Maxi
SMT-100020-NegativeBiasedWithRules		Configurator Con								
SMTTyreGroup		Group								
Aspect Ratio in Percentage	SMTLib	Family		Integer		False	False	False		
55		Feature								
65		Feature								
70		Feature								
80		Feature								
Rim Family Int type	SMTLib	Family		Integer		False	False	False		
10		Feature								
12		Feature								
14		Feature								
16		Feature								
Side walls family Int type	SMTLib	Family		Integer		False	False	False		
125		Feature								
145		Feature								
155		Feature								
Tire Diameter FreeForm type	SMTLib	Family		Floating P		False	True	False		
Tire Spec String Type	SMTLib	Family		String		False	False	False		
125/80R16		Feature								
145/70R12		Feature								
145/80R10		Feature								
155/55R14		Feature								
Unassigned Families										

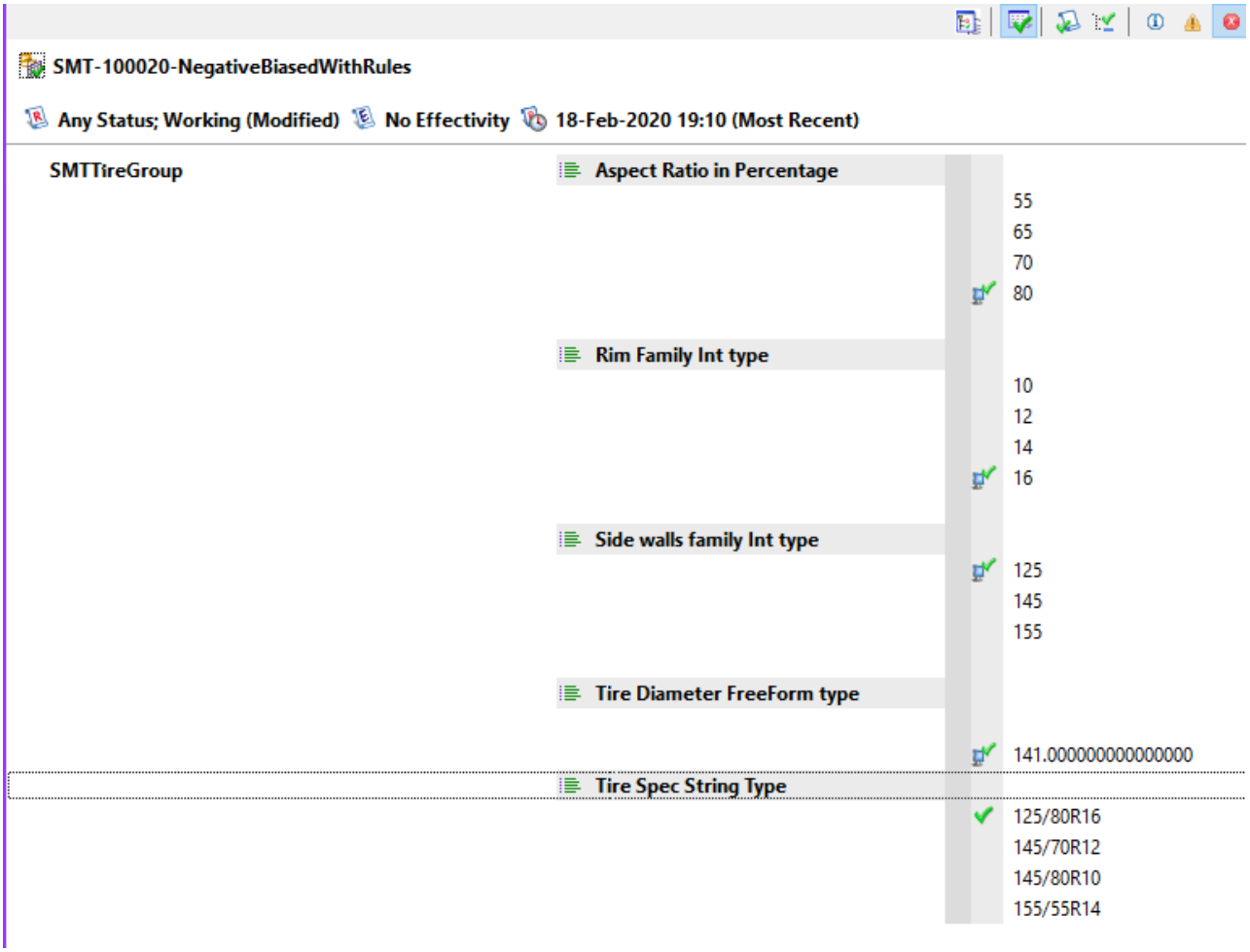
2. The following free-form rule is created to calculate the diameter of a tire.

```
(assert(=|[SMTLib]Tire Diameter FreeForm type| (+ |[SMTLib]Side walls
familty int type||[SMTLib]Rim Family Int type|)))
```



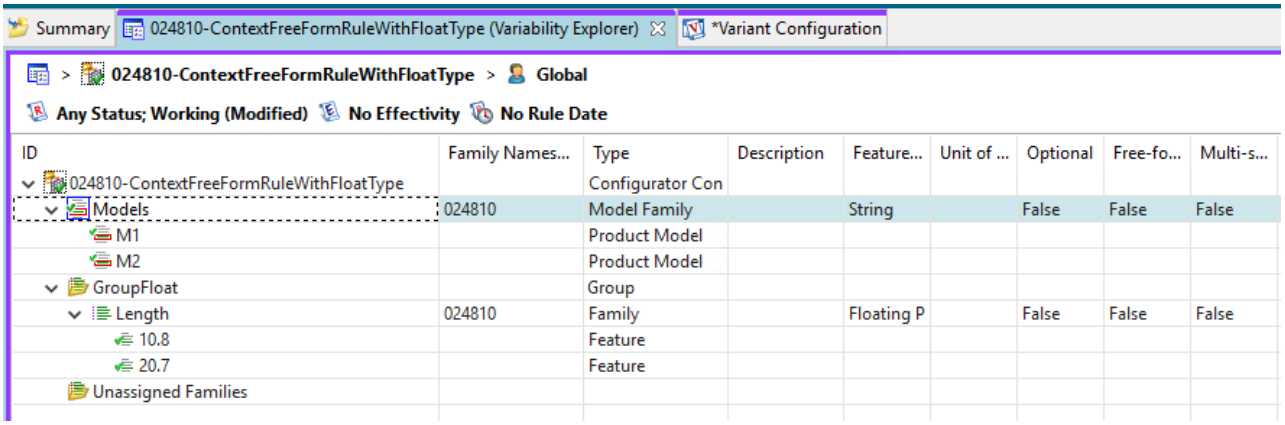
3. Open the **Variant Configuration** view to make your selections. Select the values for **Rim Family Int Type** and **Side Walls Family Int Type**, and then click **Expand**.

   The system calculates **Tire Diameter FreeForm type** based on the formula specified in the free-form rule.

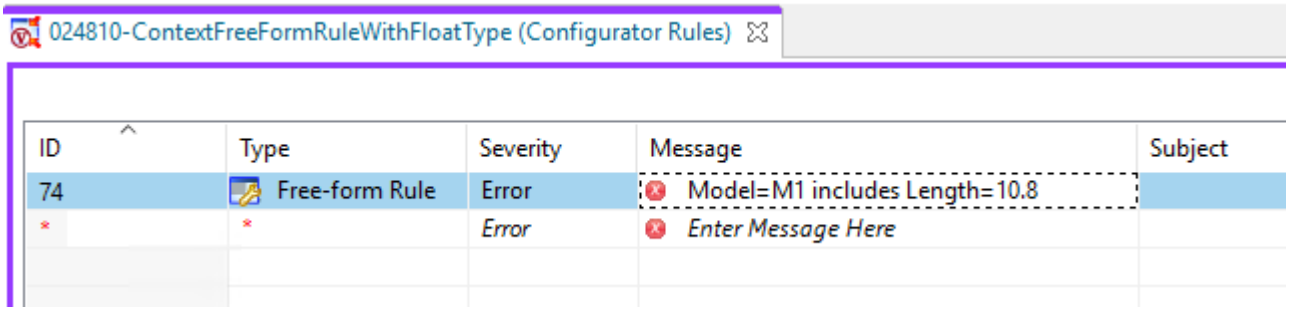


Example: Using a free-form rule with float type family

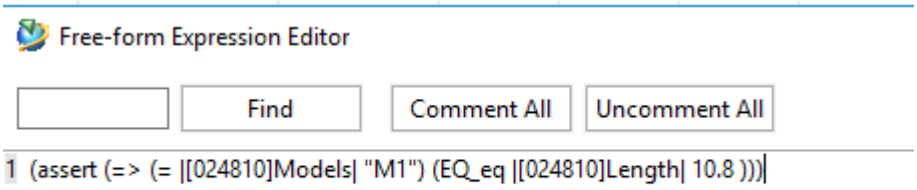
1. A positively biased configurator context is created with the following groups, families, and features.



2. The following free-form rule is created, stating that the **M1** model implies a value of **10.8** for the float family **Length**.

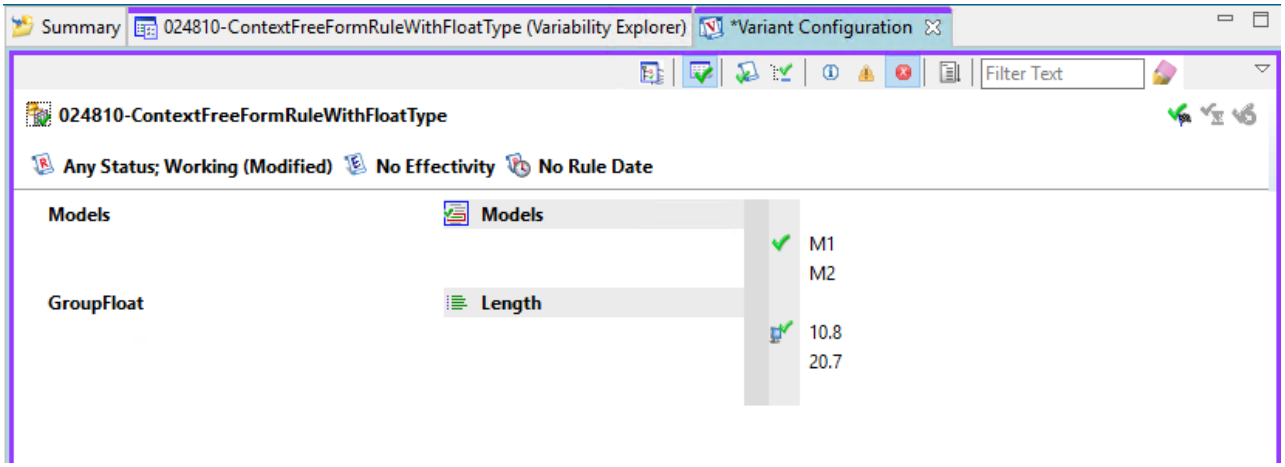


The SMTLib script is entered in the free-form expression editor.



3. Open the **Variant Configuration** view to make your selections. Select **M1** and click **Expand**.

The system selects **10.8** for the **Length** based on the free-form rule.



Validating a configuration

How do you validate a configuration?

When you create or configure and assign features to families, you can validate the expression on demand by clicking **Validate** or by using the shortcut menu. The system evaluates the constraints and returns a list of information, warnings, and errors on the various combinations.

Product Configurator on Rich Client — Usage dynamically finds and configures all relevant information. This information is stored on a number of objects that revise and configure their revisions independently with a revision rule. Teamcenter finds and configures all relevant information with respect to the configurator context.

You validate a configuration against constraints such as the following:

- Availability, inclusion, exclusion, default, and global rules.
- Feature summaries and feature packages.
- Mandatory family constraints.

  A valid and complete configuration must select a feature in every mandatory family.
- Allocation constraints.

  A valid configuration must not select a feature that is not allocated, or a feature referencing a family that is not allocated.
- Revision configuration constraints.

  For every feature in a valid configuration, the precedence rules in the revision rule must configure a revision for all of the following object types:

  - Feature
  - Feature allocation
  - Family
  - Family allocation
- Effectivity configuration constraints.

  For every feature in a valid configuration, the system must be able to configure a revision that matches the effectivity criteria of the revision rule for all of the following object types:

  - Feature
  - Feature allocation
- Product line definition constraints.

  A valid configuration that selects a product line must not violate any constraints when the product line definition is expanded to include the features listed as features of the product line and vice versa. The system observes an equivalence rule between the product line and the combination of features that belong to the product line.
- Product model definition constraints.

  A valid configuration that selects a product model must not violate a constraint when the product model definition is expanded to include the features listed as features of the product model and vice versa. The system observes an equivalence rule between the product model and the combination of included features that belong to the product model.

Teamcenter applies all of the inclusion, exclusion, availability, and default rules, including the rules that are shared across multiple configurator contexts. The system does not report any errors on shared rules if features used in applicability section are not allocated or available in the configurator context. You see a warning if features used in the rule are not allocated to the target configurator context.

When a configuration has violations, no feature expansion takes place for a rule without a violation.

> Note:
>
> Contact your system administrator if the system returns a validation failure without showing a list of information, warnings, or errors. It may be an access control issue.

Validation during authoring guarantees the verdict of validation. It provides a list of violations as an additional output.

Note:

The system stops at the first violation. If an expression has more than one set of violations, the first list of violations may not be consistent on each validation request.

Configuring and analyzing configurations

Different modes to analyze configurations

You can use the **Order** mode or the **Overlay** mode to configure and analyze the variant configuration.

- **Order** mode

  This mode is intended when the product is ready to be shipped. It is for a user who wants to quickly arrive at a 100% BOM. The 100% BOM is an *as sold* product configuration. An example of this is the configuration of a car that is going to be built and shipped to the dealer.

  You can use a 100% BOM to create a prototype, perform simulations, compute the price, calculate the weight of the product, or visualize the structure.

  In this mode, the system considers all the information that it can get to autocomplete the configuration. You can select a certain model and select the country of sale, for example, UK or India, and the system automatically selects the right-hand drive. The system scans the entire knowledge base of the configurator in order to predict what the final 100% product looks like. You can select another color or a different engine. The configurator attempts to consider your inputs and complete the configuration as much as possible as per the default rules.

  In cases where your inputs and the default rules are insufficient, the configurator does not achieve the original goal of completing a 100% configuration. When some user options are possibly left open, it achieves only a 90% configuration.

- **Overlay** mode

  This mode is intended when the product is in the development stage. It does not expect you to specify the complete configuration. You can, but it is not necessary. It configures all the features and the product content that are compatible with your inputs. For example, a new fuel filter, that you plan to introduce might have collision or interference problems within the existing space. However, at this stage, your focus is in determining all the environments in which the fuel filter can be assembled. The filter can be assembled in the 1.5 liter, 2.0 liter, and 3.0 liter gasoline engine options with or without an automatic transmission. However, the filter cannot be used in a diesel engine. Therefore, in this mode, when you select a fuel filter, you see a large overlay with various types of gasoline engines, but not a diesel engine. You can view only the options that are compatible with your selection.

Therefore, both modes converge for complete configurations, even if they might begin with very different starting positions, based on more inputs provided by the user. While the **Order** mode might automatically complete the empty input configurations to come up with a complete, default product,

the **Overlay** mode leaves the same input configuration empty. When the input expression reaches a complete product configuration, the system behavior of both modes is identical.

How do you configure and analyze configurations?

You can use the following methods to configure and analyze the variant configuration:

- Configure with custom variant configuration using manual validation.

  In manual validation mode, no wizards are used. Variant feature defaults and configuration validation are not automatically applied.

  In the manual mode, you can choose between the **Order** and the **Overlay** configuration modes.

Manual **Order configuration mode**	*Manual* **Overlay configuration mode**
Use to validate a complete and buildable configuration.	Use to validate a nonbuildable and incomplete configuration.
Selection of more than one feature from the single select family is not allowed. During validation, if you select more than one feature from the single select family, the system generates an error message.	Selection of more than one feature from the single select family is allowed.
When you click **Expand**, the discretionary families, if not assigned any value through the system rules or user input, are assigned with the symbol.	When you click **Expand**, the discretionary families, if not assigned any value through the system rules or user input, are not assigned any selection or value.
Content filtering is allowed for both **Valid and complete product configuration** and **Valid and incomplete product configuration**. If the configuration is not valid, the system displays an error and the content filtering is not performed.	There is no restriction on filtering.

  By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** configuration mode.

  Your administrator can change the default configuration mode using the **Cfg0DefaultValidationMode** preference. By default, it is set to **Order** mode.

  > Note:
  >
  > To enforce completeness checks in the **Order** configuration mode, your administrator can manually create the **Cfg0EnableOrderModeCompletenessCheck** site preference. When this

preference is set to true, the system displays an error message and the content filtering is not performed if the configuration is *valid and incomplete* or *invalid*. If the preference is not created, the system does not enforce completeness checks in the **Order** mode.

In addition, the administrator can manually create the **cfgShowCriteriaValidationWarning** site preference. When this preference is set to **true**, the system displays the content filtering warning as a pop-up message. If the value is set to **false**, the system logs the warning in the syslog and does not display it as a pop-up message. If the preference is not created, the system displays the warning with invalid criteria as a pop-up message.

Manual overlay mode allows you to select imprecise or ambiguous features for validating your configuration. However, if you switch from a manual overlay mode, while having imprecise or ambiguous selections, to an order mode, the system resets your input. As a result, you may lose your selections in your imprecise configuration.

- Configure with custom variant configuration using guided configuration.

  Guided product configurations are used to configure products by optimizing the required input. To optimize input, it allows you to prepopulate choices and at the same time provides maximum flexibility to get exactly what you and your customers want by overriding the system selections.

  In guided validation mode, a wizard-like format is used to display families of one group at a time. Variant feature defaults and configuration validation are automatically applied whenever you switch groups.

  In addition to applying the existing configurator rules that directly impact the configuration, such as setting defaults and applying availability, inclusion, and exclusion rules, Teamcenter evaluates all rules and hides any selections that become invalid. Different symbols distinguish selected features set by user from those set by the system.

  For example, an end user such as a customer, dealer, or sales engineer can go through the whole configuration process without seeing or debugging any errors or warnings. This ensures that there is always a valid configuration path available for these users based on selections your customers make.

  When you select a different group, Teamcenter does the following:

  - Validates the current selections. Only valid selections are shown.
  - Applies system selections based on the configurator rules validations to the families in the next group.
  - Applies default rules to the families in the next group.

  In the guided mode, you can choose between the **Order** and **Overlay** configuration modes.

Guided **Order configuration mode**	*Guided* **Overlay configuration mode**
The discretionary families, if not assigned any value through the system rules or user input, are assigned with the symbol by default.	The discretionary families, if not assigned any value through the system rules or user input, are not assigned any default selection or value.
The values of optional Boolean families and values of multi-select families are auto populated with the default negation value, if no value in the family was explicitly selected nor enforced by any of the configurator rules.	The values of optional Boolean families and values of multi-select families are not assigned with default selection, if not assigned any value through the system rules or user input.
Default rules in the system are applied.	Default rules in the system are *not* applied.
Low severity rules in the system are honored.	Low severity rules in the system are *not* honored.
Content filtering is allowed for both **Valid and complete product configuration** and **Valid and incomplete product configuration**. If the configuration is not valid, the system displays an error and the content filtering is not performed.	There's no restriction on filtering. Content filtering is performed with **Valid and complete product configuration** and **Valid and incomplete product configuration**.

Note:

To enforce completeness checks in the **Order** configuration mode, your administrator can manually create the **Cfg0EnableOrderModeCompletenessCheck** site preference. When set to true, the system displays an error message and the content filtering is not performed if the configuration is *valid and incomplete* or *invalid*. If the preference is not created, the system does not enforce completeness checks in the **Order** mode.

In addition, the administrator can manually create the **cfgShowCriteriaValidationWarning** site preference. When this preference is set to **true**, the system displays the content filtering warning as a pop-up message. If the value is set to **false**, the system logs the warning in the syslog and does not display it as a pop-up message. If the preference is not created, the system displays the warning with invalid criteria as a pop-up message.

Guided configuration allows you to create a valid, precise, and buildable configuration. If no feature within a mandatory family is available for a model, then the model is not buildable at all; hence, it is filtered out during guided configuration.

For example, during guided configuration, the initial screen shows only valid models. A model is considered to be valid when it has at least one feature available from every mandatory family in the configurator context.

If no valid models are available, the system generates an error message.

Manual overlay mode allows you to select imprecise or ambiguous features for validating your configuration. However, if you attempt to switch from a manual overlay mode, while having imprecise or ambiguous selections, to a guided mode, the system generates an error message. As a result, you cannot switch to a guided mode if you have an imprecise configuration.

The following are examples of imprecise configurations.

- A single select family with more than one selection.

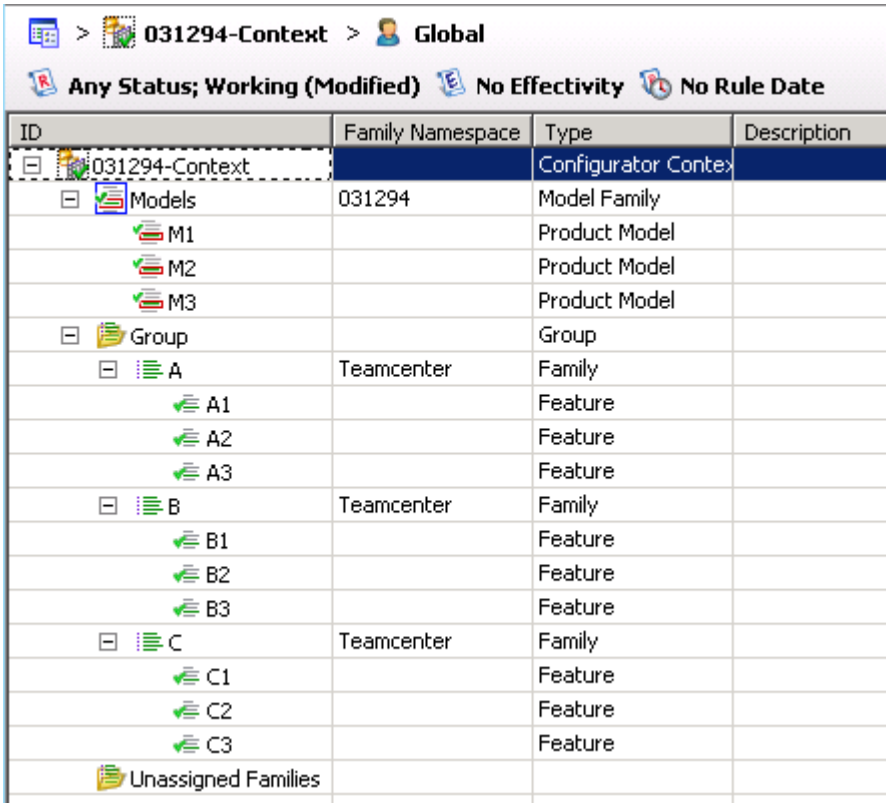
  A single select family A has three features A1, A2, and A3.

  If you select A1 and A2 for a configuration, it results in an imprecise selection.

- Single select family with selection **Family=Any**.
- Summary families with user selections.
- Selection of **!= A2**.
- Selection with branches, such as **A1 & B1 | A2 & B2**.

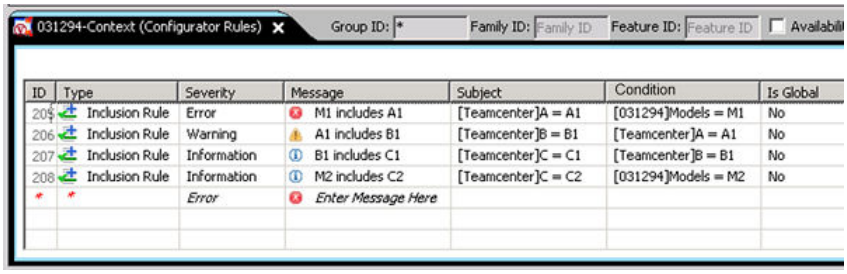
Example 1: Validating the configuration against rules with various input severity

1. A positively biased configurator context is created with the following group, families, and features.



ID	Family Namespace	Type	Description
031294-Context		Configurator Contex	
Models	031294	Model Family	
M1		Product Model	
M2		Product Model	
M3		Product Model	
Group		Group	
A	Teamcenter	Family	
A1		Feature	
A2		Feature	
A3		Feature	
B	Teamcenter	Family	
B1		Feature	
B2		Feature	
B3		Feature	
C	Teamcenter	Family	
C1		Feature	
C2		Feature	
C3		Feature	
Unassigned Families			

2. The following inclusion rules are added with different severity levels.

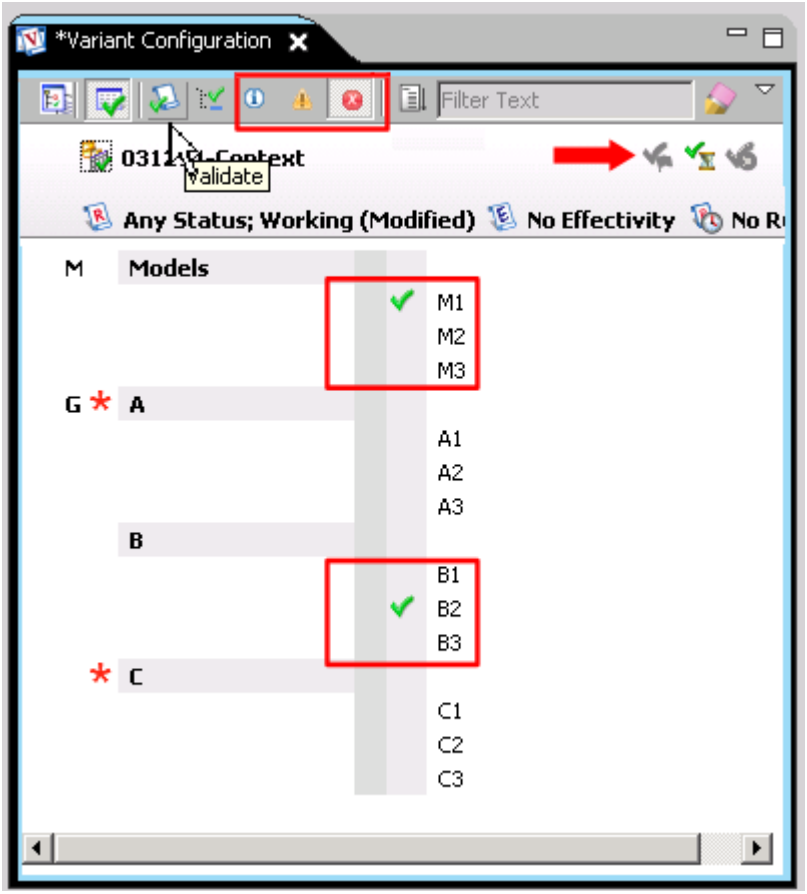


ID	Type	Severity	Message	Subject	Condition	Is Global
205	Inclusion Rule	Error	M1 includes A1	[Teamcenter]A = A1	[031294]Models = M1	No
206	Inclusion Rule	Warning	A1 includes B1	[Teamcenter]B = B1	[Teamcenter]A = A1	No
207	Inclusion Rule	Information	B1 includes C1	[Teamcenter]C = C1	[Teamcenter]B = B1	No
208	Inclusion Rule	Information	M2 includes C2	[Teamcenter]C = C2	[031294]Models = M2	No
*	*	*Error*	*Enter Message Here*			

3. The **Variant Configuration** view has the input severity set to **Error level violations** .

   By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** mode.

   M1 and **B2** are selected. When you click **Validate** , the system considers this configuration as **Valid and incomplete product configuration** .



Note that **B2** conflicts with the inclusion rule 206. However, the rule 206 has the severity set to **Warning**, and the input severity in the **Variant Configuration** view is set to **Error**. Because the **Warning** severity is ranked lower than the **Error** severity, the validation process does not consider the inclusion rule 206. During validation, the system only considered the user input and did not select any additional features.

The following is the ranking of configurator severity levels:

a. Error (highest severity)

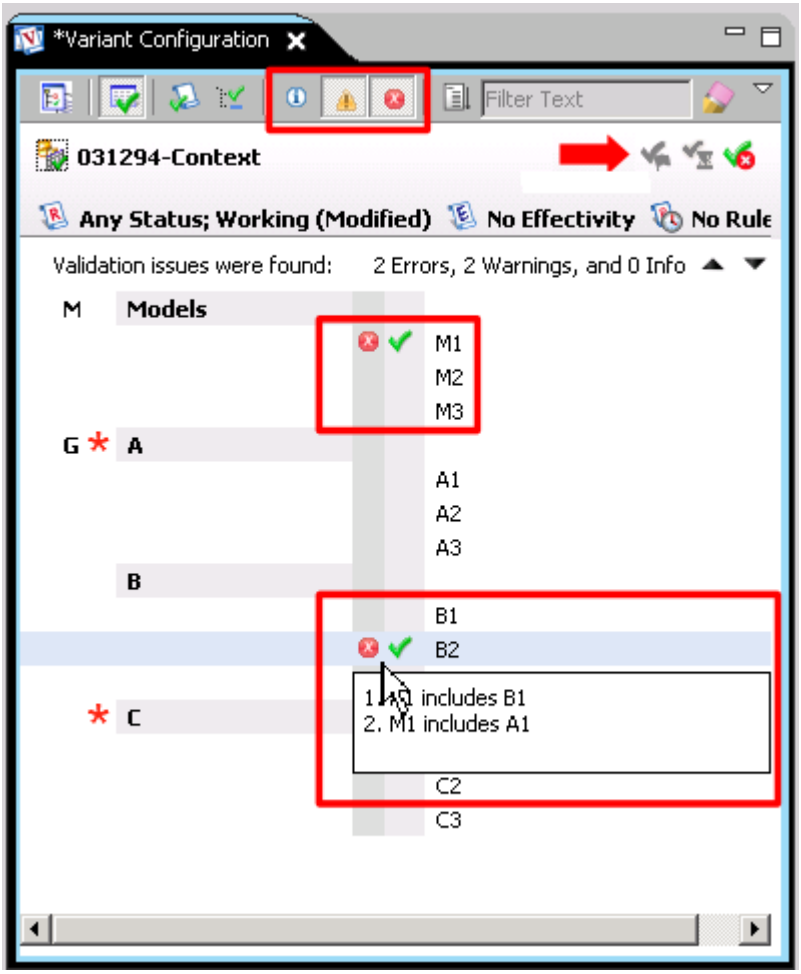
b. Warning

c. Information

d. Default (lowest severity)

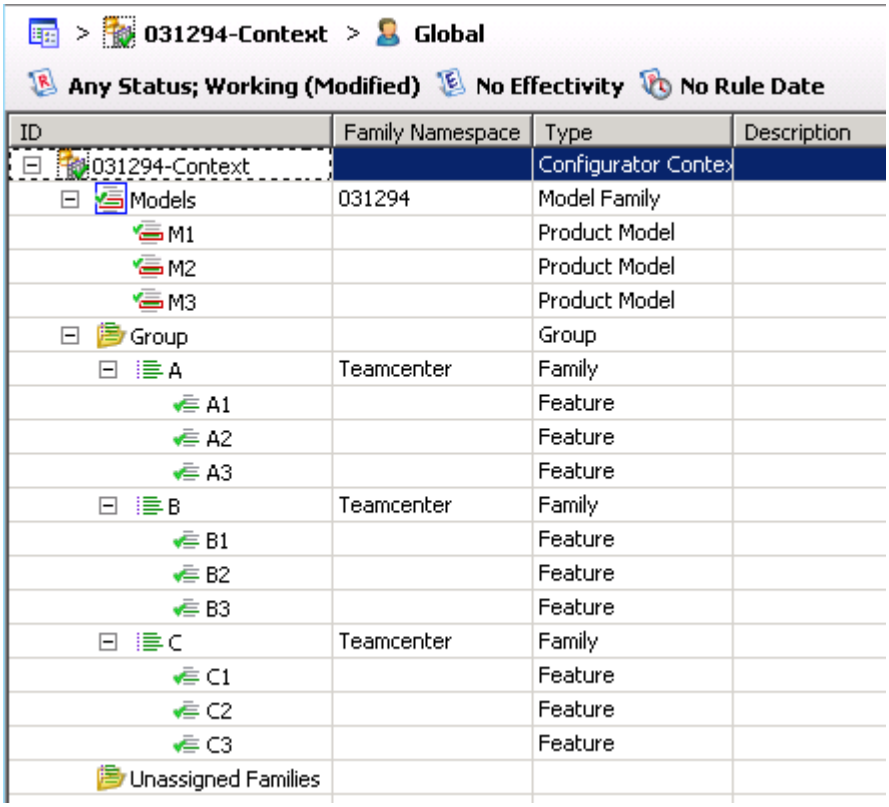
4. Next, change the input severity in the **Variant Configuration** view to both **Warning** and **Error**. As the result, the configurator rules with the **Warning** severity are also considered during validation. Violating the **Warning** severity rule results in an invalid validation.

Therefore, when you validate this configuration, the system considers it as **Invalid product configuration** because of the inclusion rule 206.



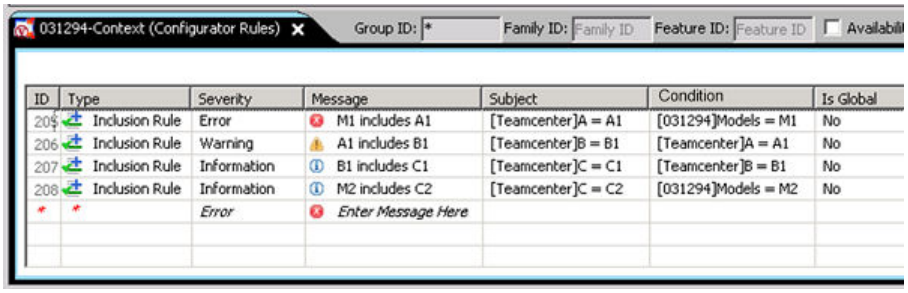
Example 2: Validating and expanding the configuration in the manual overlay mode

1. A positively biased configurator context is created with the following group, families, and features.



ID	Family Namespace	Type	Description
031294-Context		Configurator Conte>	
Models	031294	Model Family	
M1		Product Model	
M2		Product Model	
M3		Product Model	
Group		Group	
A	Teamcenter	Family	
A1		Feature	
A2		Feature	
A3		Feature	
B	Teamcenter	Family	
B1		Feature	
B2		Feature	
B3		Feature	
C	Teamcenter	Family	
C1		Feature	
C2		Feature	
C3		Feature	
Unassigned Families			

2. The following inclusion rules are added with different severity levels.

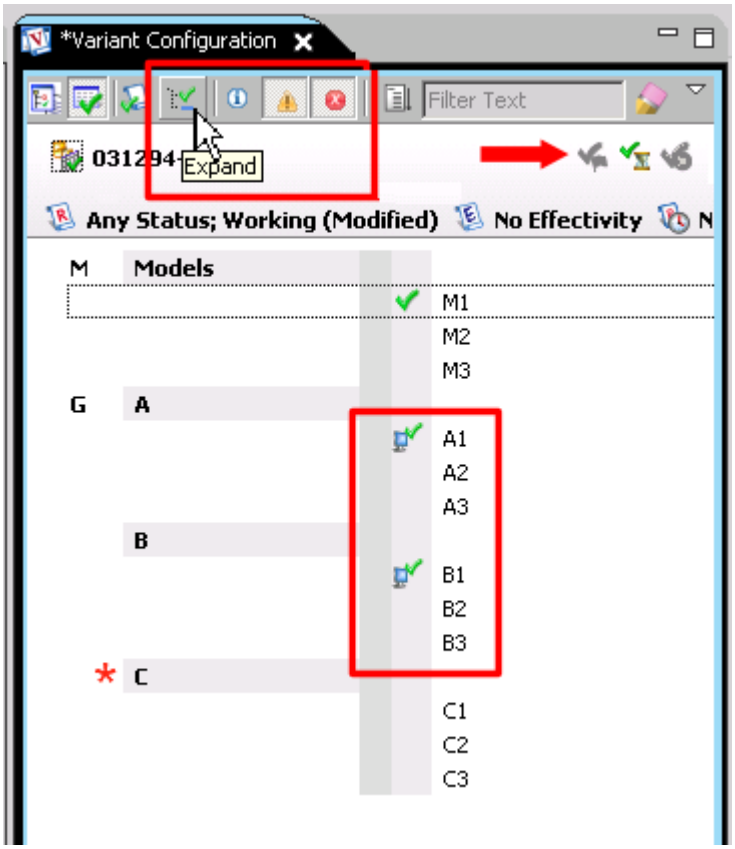


ID	Type	Severity	Message	Subject	Condition	Is Global
205	Inclusion Rule	Error	M1 includes A1	[Teamcenter]A = A1	[031294]Models = M1	No
206	Inclusion Rule	Warning	A1 includes B1	[Teamcenter]B = B1	[Teamcenter]A = A1	No
207	Inclusion Rule	Information	B1 includes C1	[Teamcenter]C = C1	[Teamcenter]B = B1	No
208	Inclusion Rule	Information	M2 includes C2	[Teamcenter]C = C2	[031294]Models = M2	No
*	*	*Error*	*Enter Message Here*			

3. The input severity in the **Variant Configuration** view is set to both **Warning** and **Error** . The configuration mode is set to **Overlay**. **M1** is selected.

   When you click **Expand** to analyze this configuration, the system considers this configuration as **Valid and incomplete product configuration** .

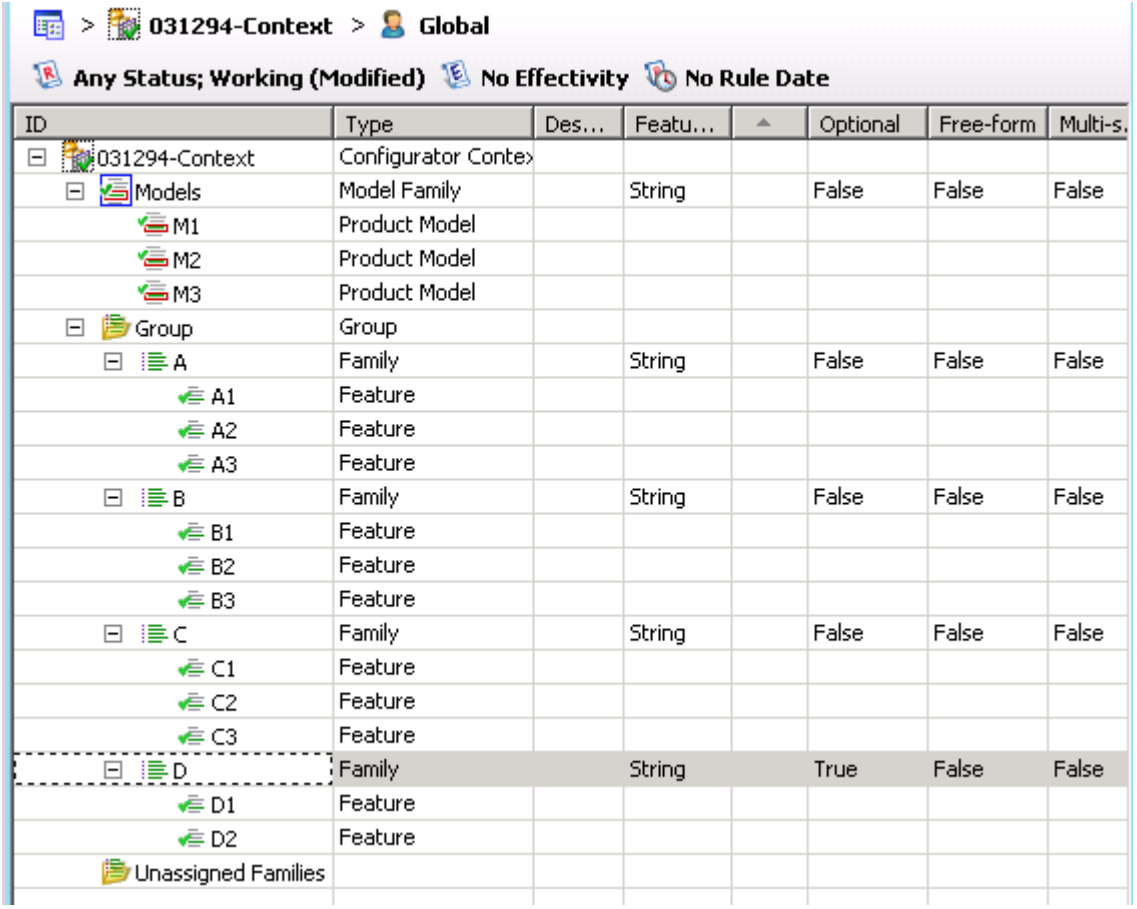
   - Both **A1** and **B1** are system-assigned and considered the only solution based on the system rules with the **Warning** and above severity.
   - Even though the include rule 207 enforces **C1** with a precondition of **B1**, **C1** is left unassigned because this rule has the **Information** severity while the input severity is set to **Warning**.



Example 3: Validating and expanding the configuration in the manual order mode

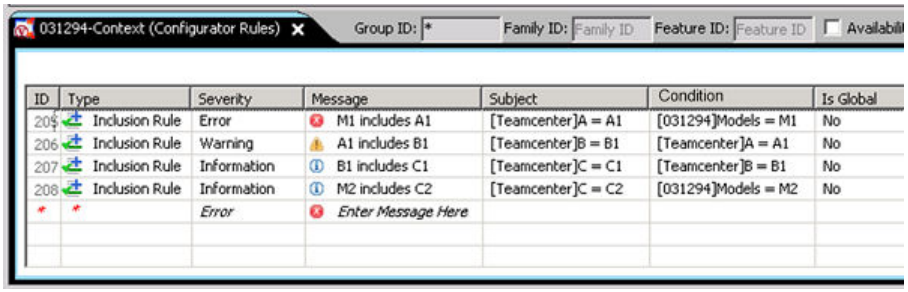
1. A positively biased configurator context is created with the following group, families, and features.

   Note that this context contains an optional family **D**.



ID	Type	Des...	Featu...		Optional	Free-form	Multi-s.
031294-Context	Configurator Contex						
Models	Model Family		String		False	False	False
M1	Product Model						
M2	Product Model						
M3	Product Model						
Group	Group						
A	Family		String		False	False	False
A1	Feature						
A2	Feature						
A3	Feature						
B	Family		String		False	False	False
B1	Feature						
B2	Feature						
B3	Feature						
C	Family		String		False	False	False
C1	Feature						
C2	Feature						
C3	Feature						
D	Family		String		True	False	False
D1	Feature						
D2	Feature						
Unassigned Families							

2. The following inclusion rules are added with different severity levels.



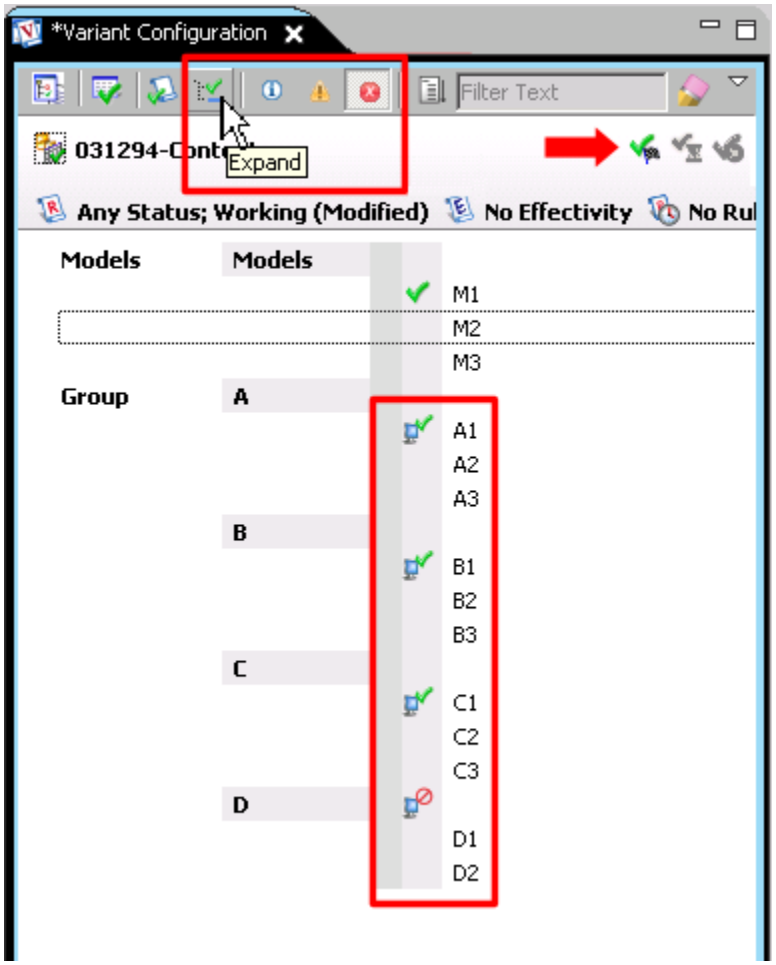
ID	Type	Severity	Message	Subject	Condition	Is Global
205	Inclusion Rule	Error	M1 includes A1	[Teamcenter]A = A1	[031294]Models = M1	No
206	Inclusion Rule	Warning	A1 includes B1	[Teamcenter]B = B1	[Teamcenter]A = A1	No
207	Inclusion Rule	Information	B1 includes C1	[Teamcenter]C = C1	[Teamcenter]B = B1	No
208	Inclusion Rule	Information	M2 includes C2	[Teamcenter]C = C2	[031294]Models = M2	No
*	*	*Error*	*Enter Message Here*			

3. The input severity in the **Variant Configuration** view is set to **Error**, by default. The configuration mode is set to **Order**. **M1** is selected.

   When you click **Expand** to analyze this configuration, the system considers this configuration as **Valid and complete product configuration**.

   - Even though the input severity is set to **Error**, the existing configurator rules with lower severity, such as warning and information, are considered. Therefore, **B1** and **C1** are selected.

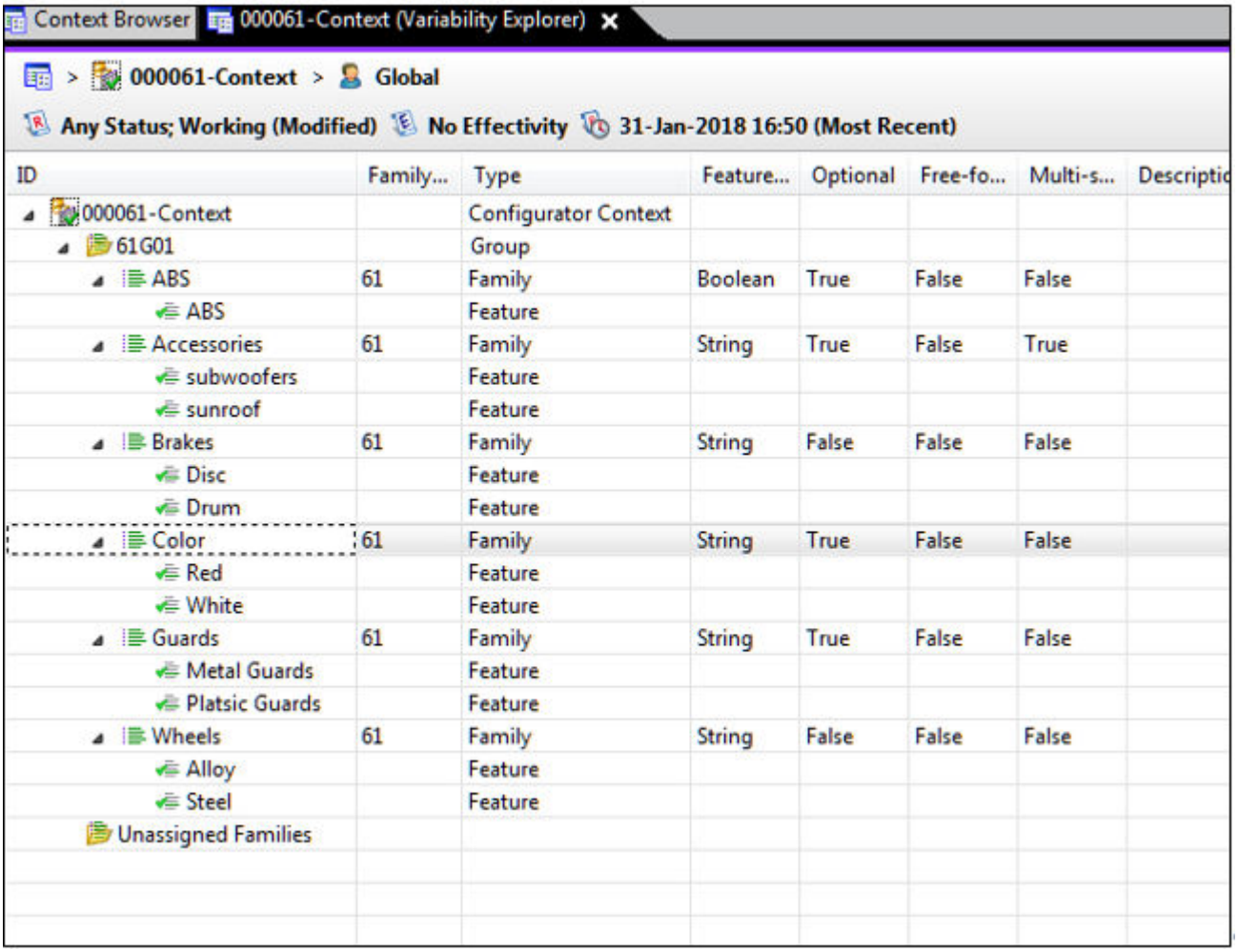
- The family **D** is assigned an empty value because it is an optional family. It indicates that no value from the family **D** is selected.



Example 4: Configuring your context in the guided mode

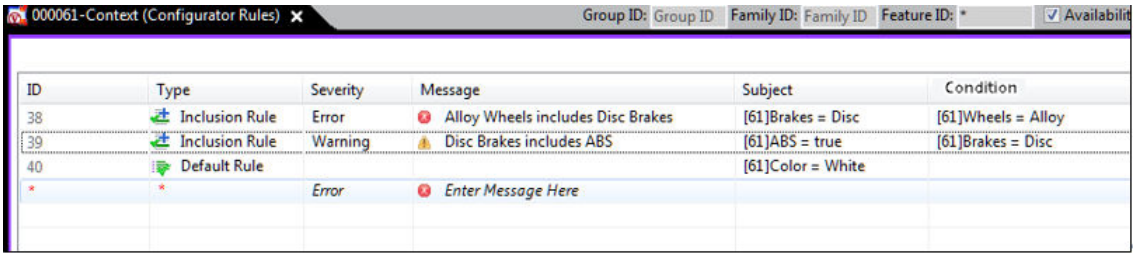
1. A positively biased configurator context is created with the following group, families, and features.

   Note that this context contains optional families.



ID	Family...	Type	Feature...	Optional	Free-fo...	Multi-s...	Descriptic
000061-Context		Configurator Context					
61G01		Group					
ABS	61	Family	Boolean	True	False	False	
ABS		Feature					
Accessories	61	Family	String	True	False	True	
subwoofers		Feature					
sunroof		Feature					
Brakes	61	Family	String	False	False	False	
Disc		Feature					
Drum		Feature					
Color	61	Family	String	True	False	False	
Red		Feature					
White		Feature					
Guards	61	Family	String	True	False	False	
Metal Guards		Feature					
Platsic Guards		Feature					
Wheels	61	Family	String	False	False	False	
Alloy		Feature					
Steel		Feature					
Unassigned Families							

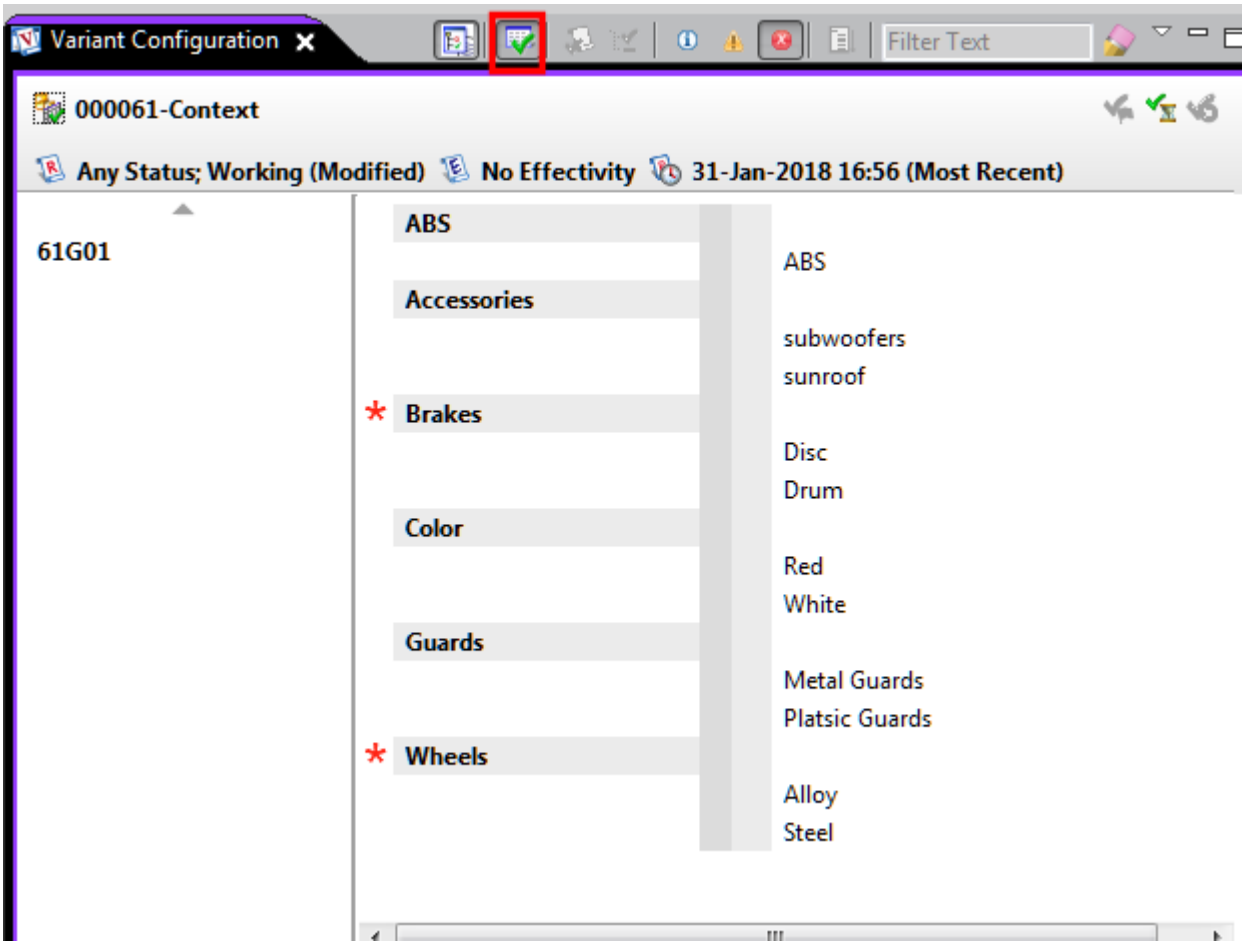
2. The following inclusion rules are added with different severity levels.



ID	Type	Severity	Message	Subject	Condition
38	Inclusion Rule	Error	Alloy Wheels includes Disc Brakes	[61]Brakes = Disc	[61]Wheels = Alloy
39	Inclusion Rule	Warning	Disc Brakes includes ABS	[61]ABS = true	[61]Brakes = Disc
40	Default Rule			[61]Color = White	
*	*	Error	*Enter Message Here*		

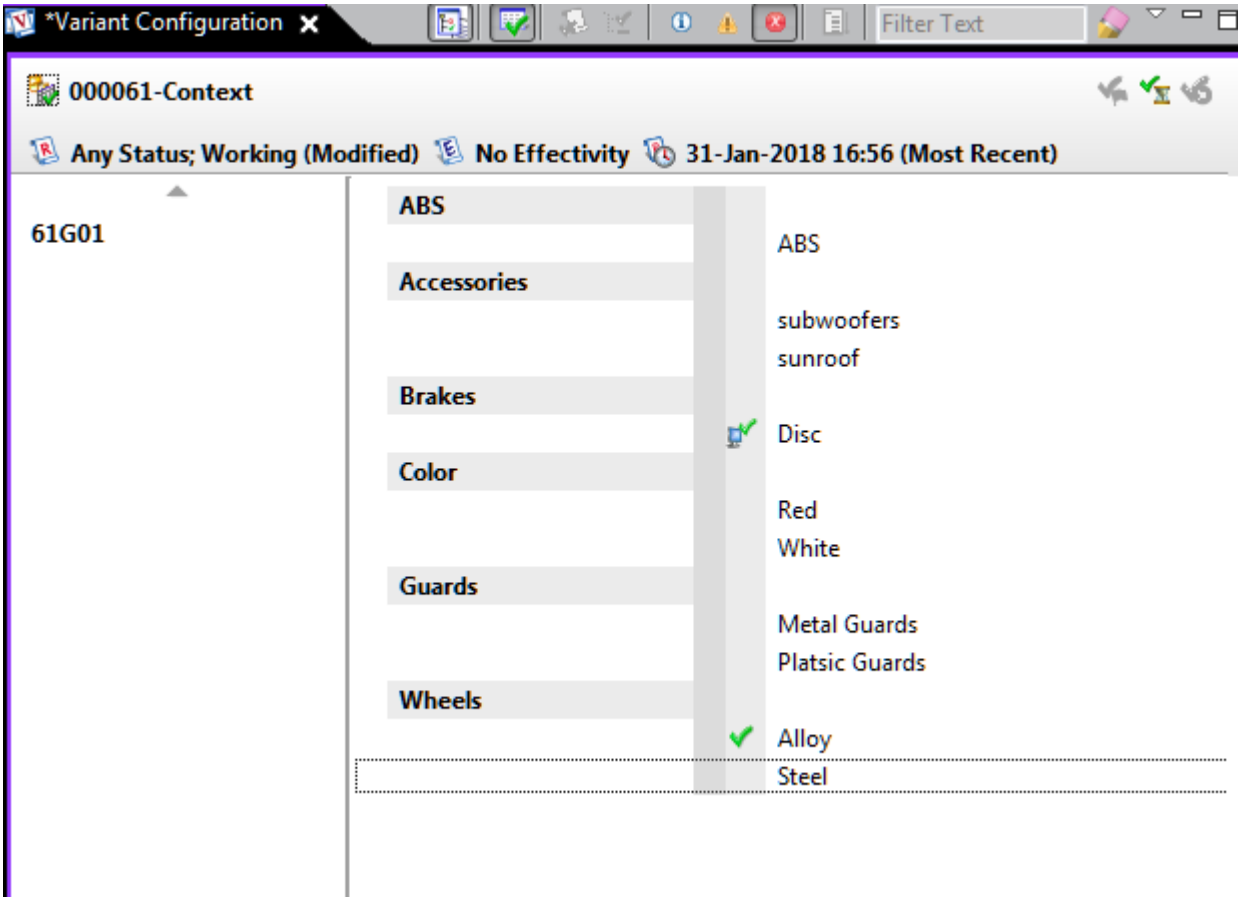
3. By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** mode.

   The input severity in the **Variant Configuration** view is set to **Error** ⊗, by default.

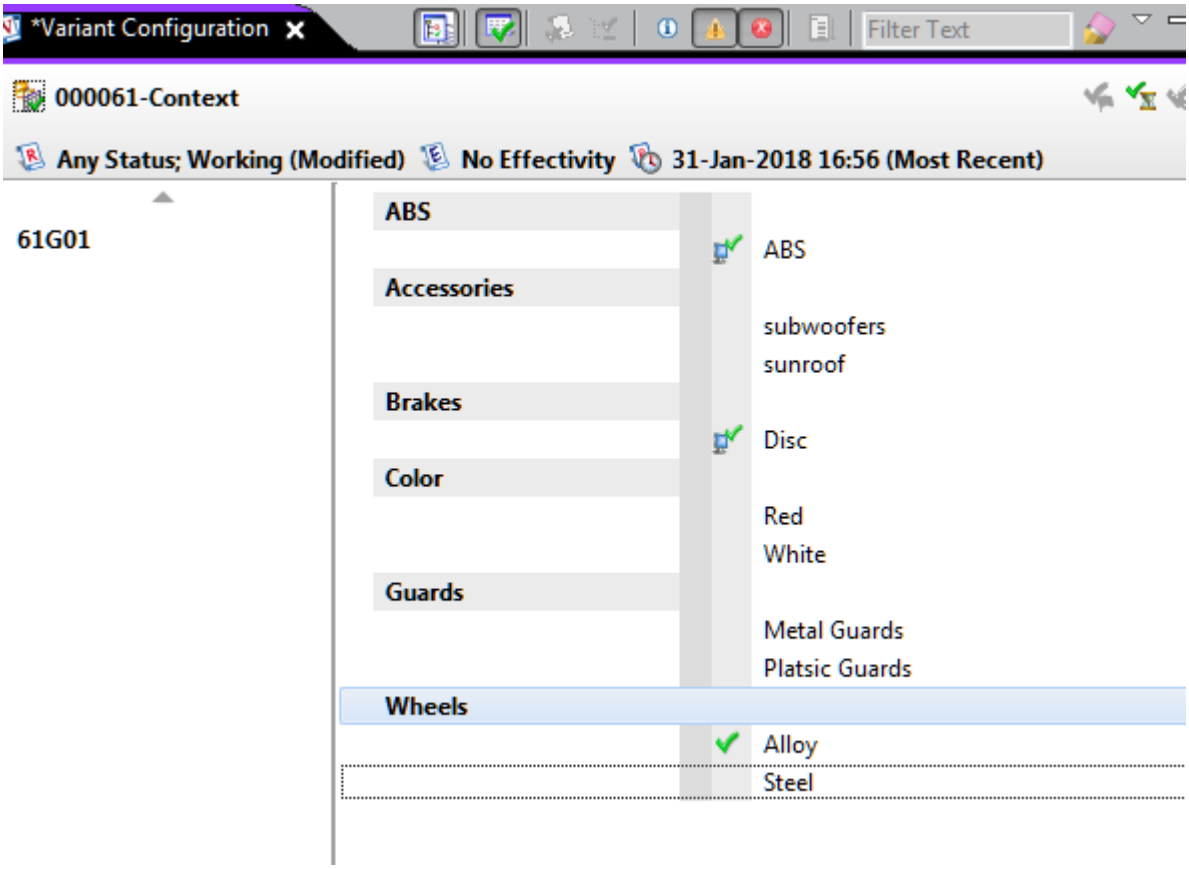


4. The **Alloy** feature is selected.

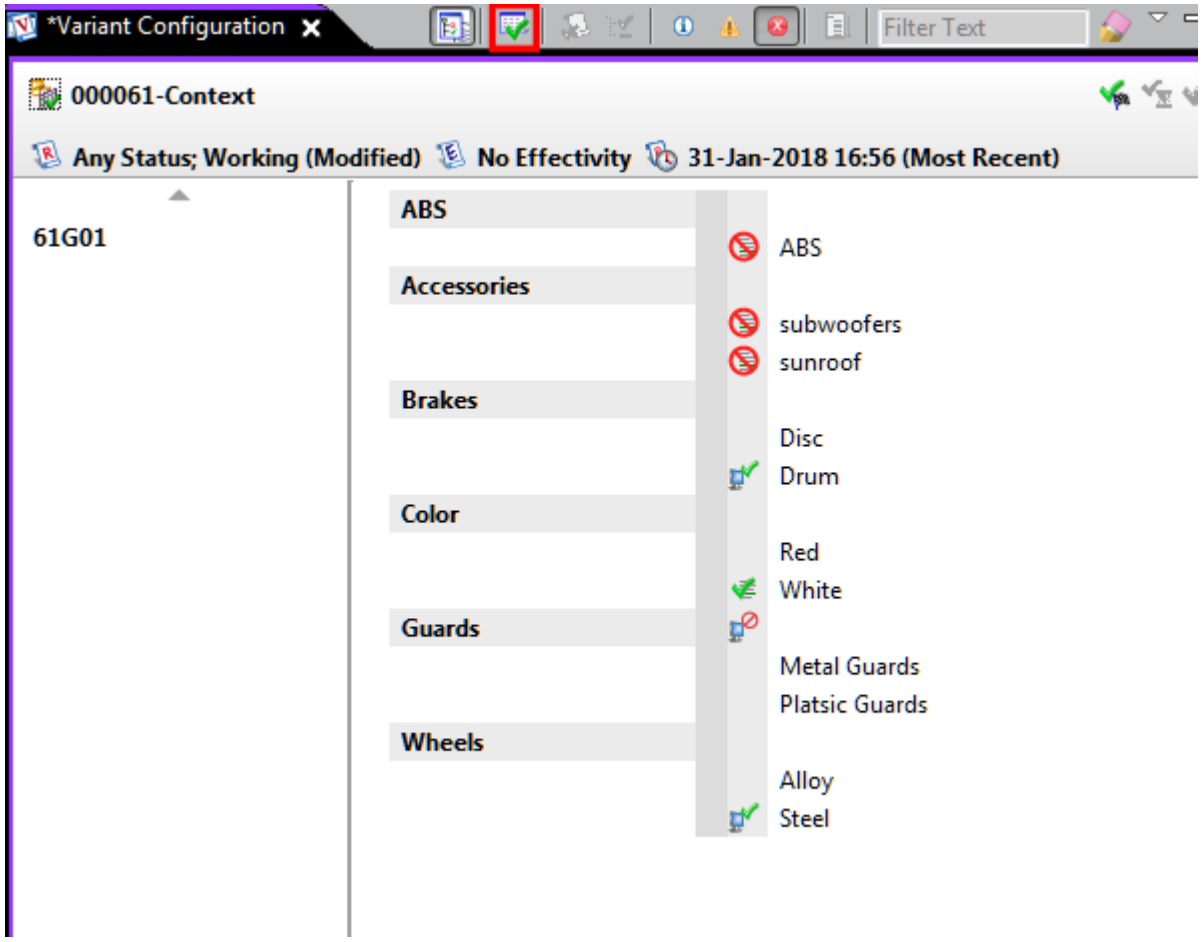
   Because you are using a guided **Overlay** mode, the system evaluates rules as per the severity set in the **Variant Configuration** view only. Currently, the input severity is set to **Error**, therefore only the inclusion rule with the **Error** severity is applied.



5. Next, change the input severity in the **Variant Configuration** view to both **Warning** ⚠ and **Error** ⊗. As a result, the configurator rules with the **Warning** severity are also considered during validation. Thus, both inclusion rules with the **Error** and **Warning** severity are applied.

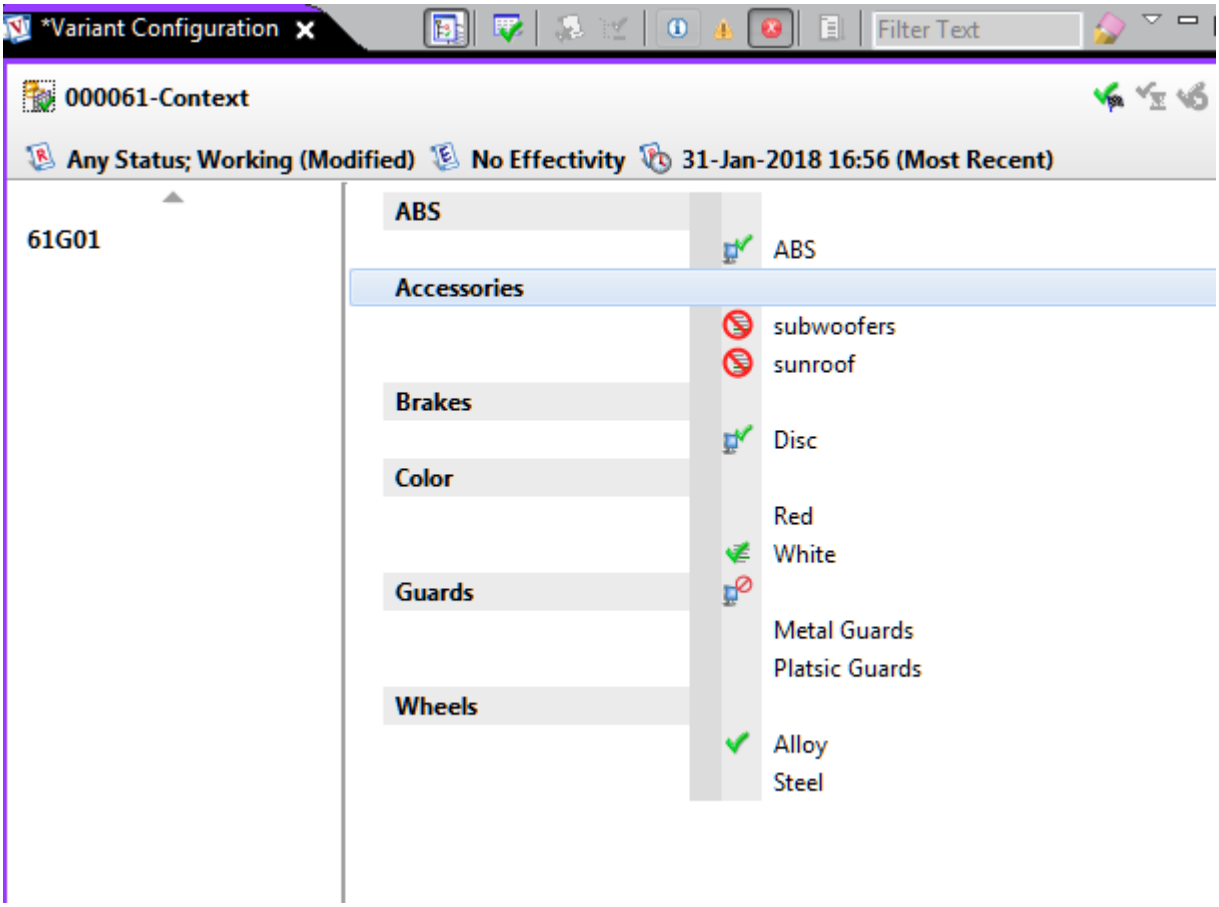


6. Now, switch the configuration mode to **Order**. As a result, the system sets default values for discretionary and multiselect families. All rules, including default rules, are evaluated.



7. The **Alloy** feature is selected.

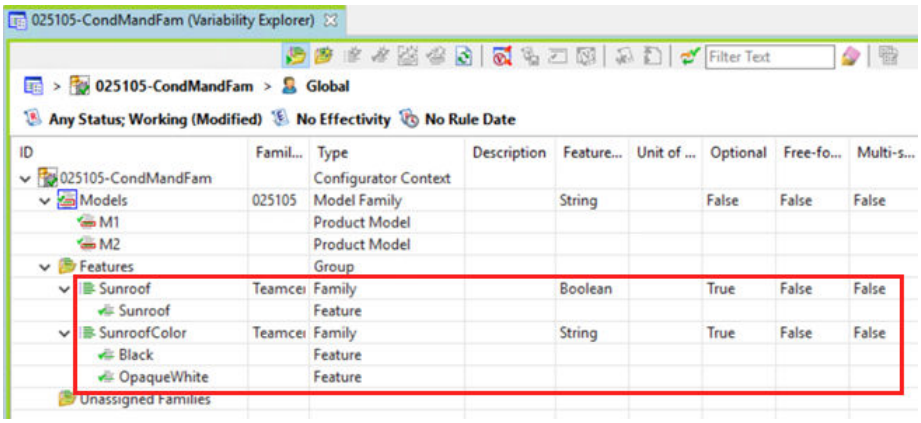
   Because you are using a guided **Order** mode, all rules are evaluated. Therefore, both inclusion rules are applied.



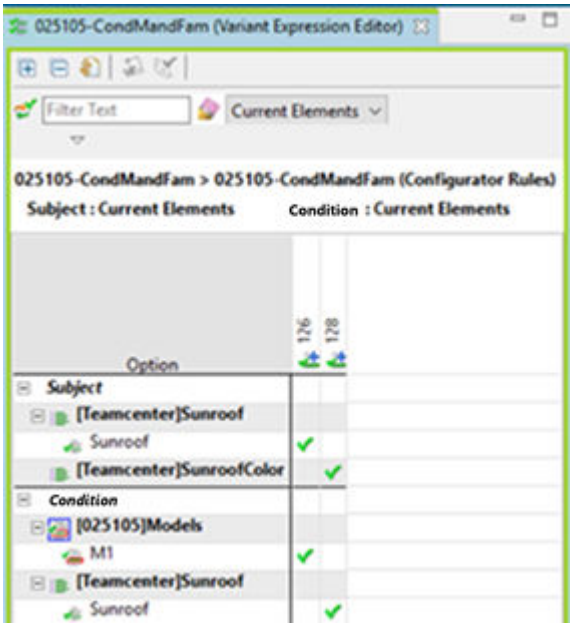
Example 5: Modeling conditional mandatory families

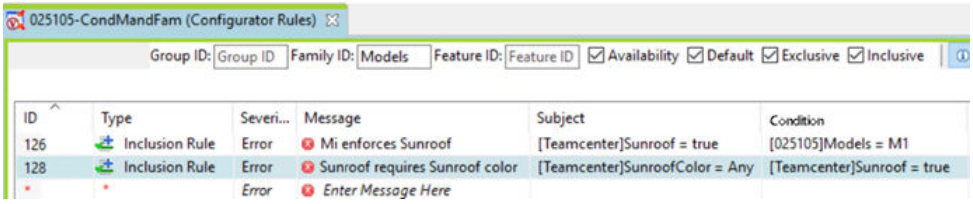
1. A positively biased configurator context is created with the following group, families, and features.

   Note that this configurator context contains an optional Boolean family and an optional string family.



2. The following inclusion rules are added.





- **Inclusion rule 126**

  The **M1** model must always have a sunroof.

- **Inclusion rule 128**

  When the **Sunroof** Boolean family is set to true, the **SunroofColor** string family becomes mandatory even though it is designated as an optional family in the system.

- **Combined effect of applying both inclusion rules**

  - The **SunroofColor** string family is mandatory for the **M1** model.

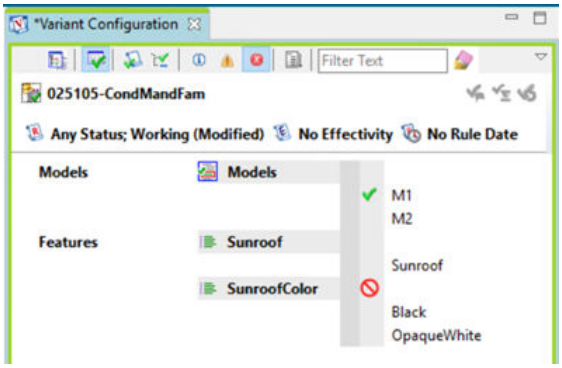
  - It may be mandatory or optional when the **M2** model is sold with or without the **Sunroof** Boolean feature, respectfully.

Note that the system apples the inclusion rule 128 when the **Sunroof** feature is set to true, either by an explicit user selection or through another applicable system rule. The **Subject** condition for the rule 128 states **SunroofColor = Any**. This means that any value allocated under the **SunroofColor** family must be true. If this condition is not met, the system considers this configuration as **Invalid product configuration** .

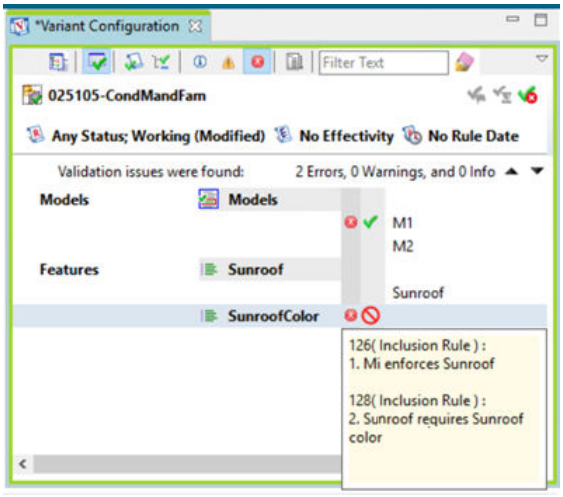
> Tip:
>
> The advantage of this rule definition is that it allows the Product Configurator administrator to add more values under the **SunroofColor** family. Without any changes to the above rule, the newly added values are also considered. The same result occurs when the administrator assigns the newly added feature to the **SunroofColor** family.

3. Next, let's validate the above rules in the **Variant Configuration** view.

   The **M1** model is selected. No color is selected for the sunroof.

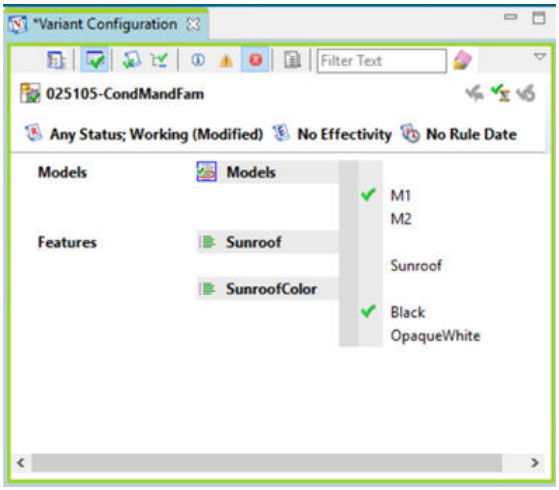


   When you click **Validate**, the system considers this configuration as **Invalid product configuration**.As the result,the system reports 2 errors.

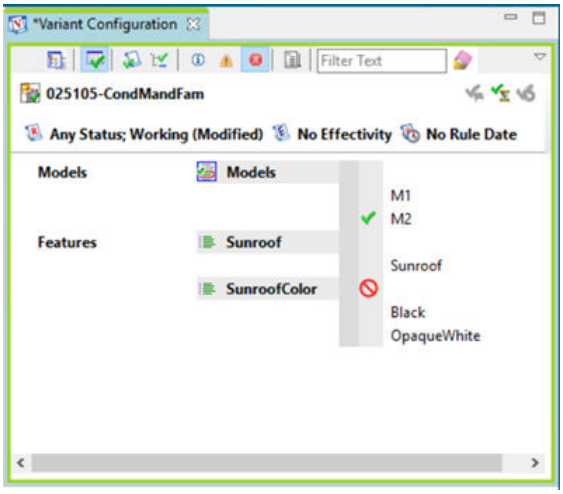


4. Next, with the **M1** model selected, let's select **Black** as the sunroof color.

   Because you selected a sunroof color for the **M1** model, the system resolves the above violations during the validation.



5. Next, let's select the **M2** model without selecting any sunroof color. When you click **Validate**, the system considers this configuration as **Valid and incomplete product configuration**. No errors are reported.



Overlaying multiple configurations for impact analysis

For some tasks, end users should only configure valid variants. If a selection violates a rule, the user is prevented from proceeding until the error condition is resolved. In other tasks, the user may be permitted to make invalid selections if they validate the selections before saving or applying the selected feature string and resolve any invalid selections. That is, you can make invalid choices during the configuration process but cannot configure an invalid variant of the structure. In some scenarios, a user may overlay multiple configurations to make comparisons or perform impact analysis.

Teamcenter allows you to:

- Overlay multiple valid variants on top of each another. All configured parts are valid for at least one single discrete valid variant, even though there is more than one variant represented in the

configuration. For example, variant rule 1 OR variant rule 2 OR variant rule 3, where each variant rule represents a valid 100% BOM product variant.

- See an unbuildable or invalid variant that includes multiple selections for a family whose features are mutually exclusive or multiple selections that are otherwise invalid for a 100% configuration. For example, `variant rule 1 = (Transmission = Manual, Engine = V6 or V8, Engine = Gasoline OR Diesel)` may configure content if `Gasoline` and `V8` are selected together, even if this is not an allowed configuration.

- Select a configuration or partial configuration, and then request to see all parts that are valid in *any* configuration together with the selected families. This configuration is often referred to as valid overlays only (VOO).

To work with overlays, you define multiple saved variant rules that can be applied simultaneously, resulting in a 120% BOM. Each time you apply a new saved variant rule or remove an applied rule, Teamcenter reconfigures the BOM.

Some of the configuration types listed are only necessary for certain user roles and tasks. You can use access control to configure whether a user can produce and save an invalid, overlayed partial, or 120% BOM variant rule, or only a complete, validated feature string.

Validating a configuration against variant constraints

When a user configures (assigns features to) families and asks Teamcenter to validate the configuration, the system evaluates the constraints and returns a list of information, warnings, and errors on the various combinations of the families.

For example:

You have a family with the variability shown in the following table and constraints of `ERROR IF L=1 AND W=20` and `ERROR IF L=2 AND W=10`.

Family	Features	Required
L	`{1; 2}`	Yes
W	`{10;20}`	Yes
O	`{Y}`	No

The validation results are as follows.

Validation result	Description	Example Configuration
Valid = true Complete = true	A valid product configuration for a product that can be built, that is, one buildable product. It is sometimes	L=1 AND W=10 AND O="" L=1 AND W=10 AND O=Y

Validation result	Description	Example Configuration
	called a 100% BOM. Configurations that assign a discrete feature to all families are complete. Empty features are considered to be discrete in this context. Only the EQUAL operator is considered to be an assignment.	
Valid = true Complete = false	A valid overlay of product configurations for multiple products that can be produced, that is, many buildable products. It is sometimes called a 120% BOM. Configurations that do not violate any constraint are valid.	L=1 AND O="" L=1 AND W=10 AND O!=Y L=1 AND W=10 AND O!="" L=1 AND W=10
Valid = false Complete = false Valid = false Complete = true	An invalid overlay of product configurations that does not match any product that can be produced, that is, no buildable products. It is sometimes called a 150% BOM.	L=1 AND W=20 L=2 AND W=10 AND O!="" L=1 AND W=20 AND O="" L=2 AND W=10 AND O=Y

You can ignore the validation check constraint messages and continue to create the variant constraints. This action may be appropriate if you are a product manager or a sales engineer who requires a new variant design based on a market need.

For example, you have the following constraint that has a family **[Production]Country** with two features, **India** and **US**.

```
Report message "You are choosing a wrong
Driver Seat position for a Car in India"
with severity "Error" when violated.
"[Production]Country = India" is not allowed
when "[Car]DriverSeatPosition = Left".
```

If you select **DriverSeatPosition = Left** and **Country = India**, the constraint returns an error. However, Teamcenter does not correct or clear the incorrect feature of **India**.

Validating a configuration against availability rules

An availability rule states the acceptability of a feature in specified conditions, which may include model conditions or a dependency on the state of other features.

For a positively biased configurator context, all features allocated to a configurator context are implicitly available for all product models within that configurator context. For a negatively biased configurator context, features are considered to be *not available* when availability rules do not exist, or if none

are effective. If a model is not mentioned, then a feature for which the availability rule is written is applicable or feasible for all models in a configurator context, if the constraints are satisfied. If an availability rule exists at the time of validating of a conflicting constraint rule, you see a validation error.

> Note:
>
> You can specify an availability bias when you create a new configurator context in the **New Business Object** dialog box. You cannot change the availability bias after the configurator context is created.

When a negative biased configurator context does not have an availability rule, then no feature is allowed in any order string or configuration. Multiple availability rules can be authored for a feature. Multiple availability rules act as a set of rules connected using a logical OR operation. The availability violation takes place if neither of the availability rules that are using the same feature are satisfied. An availability rule is not evaluated individually.

A configuration is invalid if at least one of its user or system assigned variant feature is not available.

A variant feature is available in a product configuration when no constraint rule is violated, irrespective of the type of the violated constraint, and at least one the following conditions is met:

- The configurator context is *positive* biased.
- At least one availability rule is satisfied.

A variant feature is *not* available in a product configuration if at least one of the following conditions is met:

- A constraint rule is violated, irrespective of the type of the violated constraint.
- The product context is *negative* biased and no availability rule is satisfied.

For example:

You create families with the following values.

Family	Features	Mandatory?
Model	**M1, M2**	Yes
Exterior color	**Red, Black, White**	Yes
Seat color	**Black, White, Beige**	Yes

The availability rules are set as follows.

Availability Rules	Translation for solve
Seat Color{Beige} is available for Model{M1} if Exterior Color{Red}.	*F.1* Seat Color{Beige}→Model{M1} AND (Exterior Color{Red} OR Exterior Color{White})
Seat Color{Beige} is available for Model{M1} if Exterior Color{White}.	
Seat Color{Black} is available if Exterior Color{Red} or Exterior Color{White}.	*F.2* Seat Color{Black}→Exterior Color{Red} OR Exterior Color{White}
Exterior Color{White} is available for Model{M1}	*F.3* Exterior Color{White}→Model{M1}
Exterior Color{Black} is available for Model{M1}	*F.4* Exterior Color{Black} → Model{M1} OR Model{M2}
Exterior Color{Black} is available for Model{M2}	

Examples of the validation results with a sample input order string are as follows.

Input Order String	Validation Result	Description
Model{M1} & Seat Color{Beige} & Exterior Color{Red}	Invalid	Exterior Color{Red} is not available for Model{M1}.
Model{M1} & Seat Color{Beige} & Exterior Color{Black}	Invalid	F.1 violated due to Exterior Color{Black}
Model{M1} & Seat Color{Black} & Exterior Color{White}	Valid	F.2 & F.3 are satisfied.
Seat Color{Beige} & Exterior Color{Red}	Invalid	There is no valid available rule defined for Exterior Color{Red} that passes the validation.
Model{M2} & Seat Color{Beige} & Exterior Color{Red}	Invalid	F.1 violated due to Model{M2}.
Exterior Color{White}	Valid	Due to F.1 and F.2, an assumption is that Model{M1} and Seat Color{Beige}.

Validating a configuration against variant constraints with feature packages

Creating feature packages allows you to group a set of features for convenience of selection within configuration order strings. You assign existing features as members of a package. The assigned features are connected using a logical AND operation in the variant expression.

When validating a configuration with feature packages, the AND combination of all the package members of selected feature packages is considered when processing the constraints and availability rules. If a validation rule is violated based on the feature packages, the feature package is flagged with a violation.

Members of a product model are implicitly available to the respective product model. If a feature package is a member of a product model, then members of that feature package are also implicitly available to the respective product model.

For example:

You create a family with the following features.

Family	Features	Required
L	`{1; 2}`	Yes
W	`{10;20}`	Yes
H	`{a;b;c}`	Yes
O	`{Y}`	No

You create a package family with the following feature packages.

Package family	Feature Packages	Members
P	`{P1; P2}`	P1: `{L=1;W=20}` P2: `{L=2;W=10}`

You create the following constraints:

- `ERROR IF H=a AND P=P1`
- `ERROR IF H=b AND P=P1`

The validation results are as follows.

Validation result	Description	Example Configuration
Valid = true Complete = true	A valid product configuration for a product that can be built, that is, one buildable product. It is sometimes called a 100% BOM. Configurations that assign a discrete feature to all families are complete. Empty features are considered	H=c AND P=P2 AND O="" H=c AND P=P1 AND O=Y

Validation result	Description	Example Configuration
	to be discrete in this context. Only the EQUAL operator is considered to be an assignment.	
Valid = true Complete = false	A valid overlay of product configurations for multiple products that can be produced, that is, many buildable products. It is sometimes called a 120% BOM. Configurations that do not violate any constraint are valid.	L=1 AND O="" H=c AND W=10 AND O!=Y P=P2 AND O!="" P=P1
Valid = false Complete = false	An invalid overlay of product configurations that does not match any product that can be produced, that is, no buildable products. It is sometimes called a 150% BOM.	H=a AND P=P1 H=b AND P=P2 AND O!=""
Valid = false Complete = true		H=a AND P=P1 AND O="" H=b AND P=P2 AND O=Y

Validating a configuration against variant constraints with feature summaries

You can create *feature summaries* to summarize multiple features within the same family. Existing features are assigned as members of a feature summary and are connected using an exclusive OR operation in the variant expression.

Feature summaries are only available when configuring with a custom variant configuration using manual validation and *not* available using guided validation. An exclusive OR combination of all members of selected feature summaries is considered when processing constraints and availability rules. If a validation rule is violated based on a feature summary, the feature summary is flagged with the violation.

For example:

You create a family with the following features.

Family	Features	Required
L	`{1; 2}`	Yes
W	`{10;20}`	Yes
H	`{a;b;c}`	Yes
O	`{Y}`	No

You create a summary family with the following feature summaries.

Package family	Feature summaries	Members
S	{S1; S2}	S1: {L=1;L=2} S2: {W=10;W=20}

You create the following constraints:

- ERROR IF H=a AND S=S1
- ERROR IF H=b AND S=S2

The validation results are as follows.

Validation result	Description	Example Configuration
Valid = true Complete = true	A valid product configuration for a product that can be built, that is, one buildable product. It is sometimes called a 100% BOM. Configurations that assign a discrete feature to all families are complete. Empty features are considered to be discrete in this context. Only the EQUAL operator is considered to be an assignment.	H=c AND L=1 AND W=10 AND O="" H=a AND L=2 AND W=20 AND O=Y
Valid = true Complete = false	A valid overlay of product configurations for multiple products that can be produced, that is, many buildable products. It is sometimes called a 120% BOM. Configurations that do not violate any constraint are valid.	H=a AND S=S2 AND O="" H=b AND S=S1 AND O!=Y L=1 AND O!="" S=S1
Valid = false Complete = false Valid = false Complete = true	An invalid overlay of product configurations that does not match any product that can be produced, that is, no buildable products. It is sometimes called a 150% BOM.	H=a AND S=S1 H=b AND S=S2 AND O!="" H=a AND S=S1 AND O="" H=b AND S=S2 AND O=Y

Understanding variant solve precision limits

- Viewing value ranges in normalized rich client grid displays

  The rich client displays expressions created in it in the same format in which they are authored. However, if you use an expression created in another application, the rich client may request a

Teamcenter Normal Form (TNF) normalization. If so, Teamcenter formats value ranges, parentheses nesting levels, and other displayed data according to the needs of the rich client's grid. For example, it may reformat open (**min<val<max**) and closed (**min<=val<=max**) value ranges:

Type	1..10	1..	..10
Integer/Text	**>=1 & <=10**	**>=1**	**<=10**
Double/Date	**>=1 & <10**	**>=1**	**<10**

Consequently, if another client such as NX creates the integer expression **INT < 10**, the rich client displays **..9**, which is equivalent to **INT <= 9**.

- Checking expressions can be satisfied

  Teamcenter checks expression can be satisfied with the following mappings, where:

Operator	Meaning
==	Is identical to
=~	Satisfies
!~	Violates

> Note:
>
> Date ranges that include the end date using **<=** represent the time interval until the end of the timestamp specified in the date value. You set the **TC_Fnd0Booleansolve_EffectivityDateRangeFromTo** preference to specify the display format you want to use for effectivity date ranges.

- **LESS THAN** operator

  Integer

  ```
  "INT < 4.000000000000013" == "INT <= 4"
  "INT < 4.000000000000007" == "INT <= 3"
  "INT < 4"                 == "INT <= 3"
  ```

  Floating point

  ```
  "DBL < 4.5" !~ "DBL = 4.499999999999993"
  "DBL < 4.5" =~ "DBL = 4.499999999999987"
  ```

  Date

```
"Date < 00:00:02+01:00"  (2 seconds past midnight MET)
  == "Date < 00:00:02.0+01:00" (2.0 seconds past midnight MET)
  "Date < 00:00:02.5+01:00" (2500 milliseconds past midnight MET)
== "Date < 00:00:03.0+01:00" (3.0 seconds past midnight MET)
```

- **GREATER THAN OR EQUAL TO** operator

  Integer

  ```
  "INT >= 4.000000000000013" == "INT >= 5"
  "INT >= 4.000000000000007" == "INT >= 4"
  "INT >= 4"                 == "INT >= 4"
  ```

  Floating point

  ```
  "DBL >= 4.5" !~ "DBL = 4.499999999999987"
  "DBL >= 4.5" =~ "DBL = 4.499999999999993"
  ```

  Date

  ```
  "Date > 00:00:02+01:00"  (2 seconds past midnight MET)
    == "Date > 00:00:02.0+01:00" (2.0 seconds past midnight MET)
    "Date > 00:00:02.5+01:00" (2500 milliseconds past midnight MET)
  == "Date > 00:00:02.0+01:00" (2.0 seconds past midnight MET)
  ```

- **LESS THAN OR EQUAL TO** operator

  Integer

  ```
  "INT <= 4"        == "INT <= 4"
   "INT <= 3.999999999999993" == "INT <= 4"
  "INT <= 3.999999999999987" == "INT <= 3"
  ```

  Floating point

  ```
  "DBL <= 4.5" !~ "DBL = 4.500000000000013"
   "DBL <= 4.5" =~ "DBL = 4.500000000000007"
  ```

  Date

  ```
  "DATE <= 2013-03-29T08:00:00+01:00"
    == "DATE < 2013-03-29T08:00:01.0+01:00"
  ```

- **GREATER THAN** operator

  Integer

```
"INT > 4"       == "INT >= 5"
"INT > 3.999999999999993" == "INT >= 5"
"INT > 3.999999999999987" == "INT >= 4"
```

Floating point

```
"DBL > 4.5" !~ "DBL = 4.500000000000007"
   "DBL > 4.5" =~ "DBL = 4.500000000000013"
```

Date

```
"DATE > 2013-03-29T08:00:00+01:00"
  == "DATE >= 2013-03-29T08:00:01.0+01:00"
```

- **EQUAL** operator (non-empty values)

  Integer

  ```
  "INT = 4.000000000000013" == "FALSE"
  "INT = 4.000000000000007" == "INT = 4"
   "INT = 4"                == "INT = 4"
  "INT = 3.999999999999993" == "INT = 4"
  "INT = 3.999999999999987" == "FALSE"
  ```

  Floating point

  ```
  "DBL = 4.5" == "4.5 <= DBL < 4.50000000000001"
  ```

  Date

  ```
  "Date = 00:00:02+01:00"  (2 seconds past midnight MET)
    == "Date < 00:00:03.0+01:00 & Date >= 00:00:02.0+01:00"
    (the time between 2 and 3 seconds past midnight MET)
   "Date = 00:00:02.5+01:00" (2500 milliseconds past midnight MET)
    == "Date < 00:00:03.0+01:00 & Date >= 00:00:02.0+01:00"
    (the same time between 2 and 3 seconds past midnight MET)
  ```

- **NOT EQUAL** operator (non-empty values)

  Integer

  ```
  "INT != 4.000000000000013" == "TRUE"
  "INT != 4.000000000000007" == "INT != 4"
  "INT != 4"                 == "INT != 4"
  ```

- Evaluating discretionary logical families

Logical families are inherently discretionary. Expressions using logical families evaluate to **TRUE** if (and only if) the criteria match the setting **[DICT]LogicalFamilyName = LogicalFamilyName**. If the criteria contradict **[DICT]LogicalFamilyName = LogicalFamilyName**, the condition evaluates to **FALSE**. It is not important *how* the criteria contradict this setting, for example, because the setting is **[DICT]LogicalFamilyName != LogicalFamilyName** or **[DICT]LogicalFamilyName = ""**.

- Creating free-form logical families:

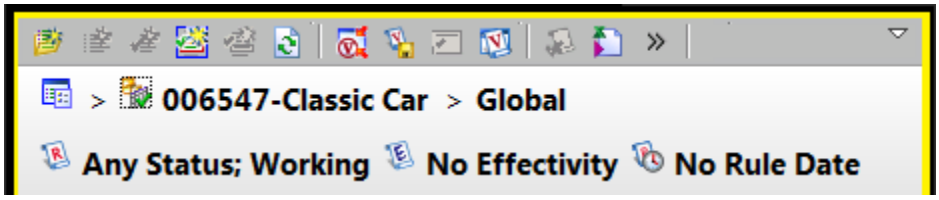
  Creating a family of free-form values implies that content authors cannot anticipate the actual value consumers will select to configure their data. Consequently, the variant condition operators **EQUAL** and **NOT EQUAL** are not meaningful. Conversely, **EQUAL** and **NOT EQUAL** are the only meaningful operators for expressions using values in logical families, in which the concepts of family and value mean the same. Teamcenter locks the list of values in logical families to a single value that matches the family name. The rich client blocks the add value operation because the user perceives the value and the family as identical.

- Creating multiselection logical families:

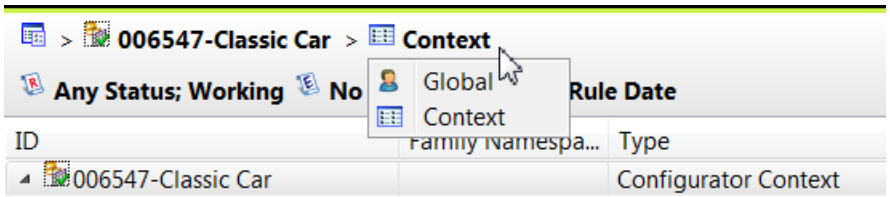
  Creating a family for multiselection values implies that there is more than one value. However, for logical families, Teamcenter locks the list of values in logical families to a single value that matches the family name. The rich client blocks the add value operation because the user perceives the value and the family as identical. Consequently, Teamcenter does not allow the creation of logical families for multiselection values.

Configuring your content

To ensure that you see only relevant and available features, you can configure your configurator context or dictionary in a variety of ways, such as by actual and configurator allocation objects, revision rules, effectivity, and rule date.



By actual and configurator allocation objects



You can choose to view all features, or only the available or allocated ones within your configurator context.

Select one of the following:

- **Global**

  Teamcenter displays actual objects representing a family or feature.

- **Context**

  Teamcenter displays configurator allocation objects for the group, family, or features, including summary and package family and features.

Button	Configurator allocation object
	Group
	Family
	Feature
	Package family
	Feature package
	Summary family
	Feature summary

By revision rule

Product Configurator supports all revision rules that contain one or more of the following entry types:

- **Status**
- **Working**
- **Unit**
- **Historic configuration date**

> Note:
>
> Revision rules with **Unit** entries are supported for backward compatibility. It is recommended to set the unit or unit range using the effectivity dialog.

By effectivity

By **setting effectivity**, you can configure variability according to specified dates.

Date and unit effectivity, as well as historic configuration date settings, are not persistent. Instead, they are typically applied at run time when the configurator context is displayed.

> Note:
>
> If a revision rule was saved with a unit setting, then it is used as unit effectivity unless it is overwritten with the effectivity dialog. If the effectivity is removed in the effectivity dialog, the underlying persistent revision unit setting becomes active.

By rule date

By setting a revision **Rule Date**, you can configure design data by date.

> Note:
>
> If a revision rule, effectivity, or rule date is changed after the user has performed selections or validations, all system selections, user selections, violations, and completeness status, if any, are cleared.

Using revision rules in Product Configurator

You can apply the out of the box **revision rule** to filter configurator data for authoring. You can only select from a fixed list of pre-authored revision rules.

An object revision must be effective to be considered as a candidate revision to be selected. A revision rule entry skips a revision if its effectivity condition does not match the effectivity criteria defined for the revision rule. If no other alternative revision exists that matches the entry, the revision rule skips to the next entry.

At a minimum, a revision requires read access to be considered as a candidate revision to be selected. A revision rule entry skips a revision if applicable access rules deny read access. If no other alternative revision exists that matches the entry, the revision rule skips to the next entry.

When you click the revision rule hyperlink, Teamcenter displays the dropdown list of revision rules available in the system.

All revision rules supported by the precedence filter are compatible with Product Configurator.

Note:

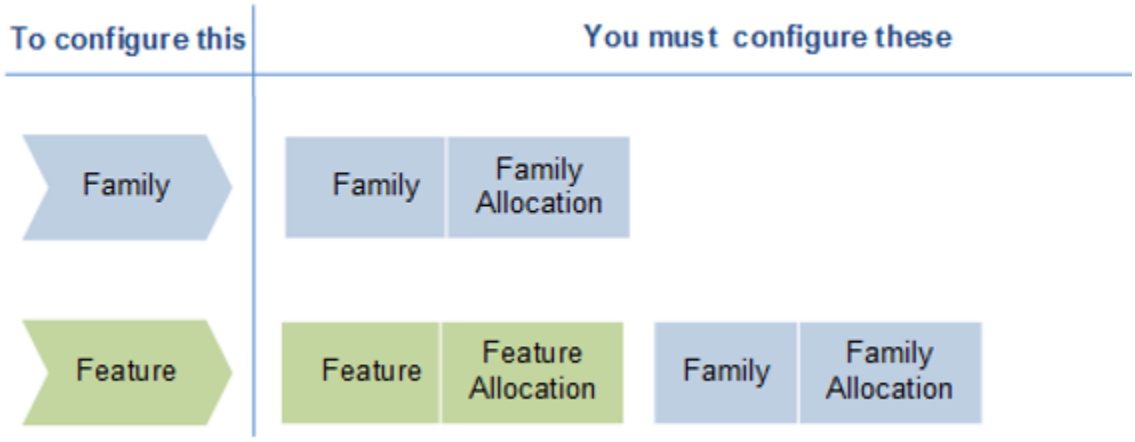
If a revision rule is saved with the **Revision Rule Suppressed = true** property value, then this rule is also filtered out.

Configuring data based on revision rules

Note:

If you plan to use revision rules to configure only released feature revisions, you must also release the configurator allocation and family objects.

Below is the chain of configurator objects that must be configured in order to show the selected object in the context or during solve, based on a selected revision rule.



For example, for a variant *feature* revision to be configured for the given revision rule, all of the following objects must be also configured:

- *Feature* revision
- *Feature allocation* revision
- *Family* revision
- *Family allocation* revision

Note:

Both the group revision and the group allocation revision do not have to be configured for the revision rule. However, if the groups are not allocated, then their families and features are shown in the **Unassigned** folder.

Working with revision rule dates in Product Configurator on Rich Client — Usage

By setting a revision rule date, you can configure historic data in Product Configurator on Rich Client — Usage by date. For example, you are a manufacturing engineer and have consumed design data on a specific date and you are now ready to consume newer versions of data. You would like to first display what was consumed in the past. By setting the revision rule date, you can display the earlier design data.

Configurator objects can be configured using revision rule dates.

You can set the revision rule date by setting a rule date in the **Variability Explorer** and the **Variant Configuration** views.

Note:

If there are multiple **Working** revisions to choose from, the system selects the one with the more recent **Create** date.

Using a configurator snapshot

Improving the system response time for the configuration requests using a snapshot mechanism

For each solve, Teamcenter collects all of the relevant configurator data before processing. However, the collection of all of the relevant configurator data may take time depending on the environment setup, the size of the configurator data and its complexity. To improve the response time for the configuration requests, Teamcenter allows you to use an internal snapshot of the configurator data for a specific configuration. Then, Teamcenter reuses it if downstream users ask for the same configuration, for a number of times. The configurator snapshot enables the downstream users to work on a constant or frozen configurator data and experience a better response time.

You work on a snapshot or a frozen set of the configurator data and use that data to get consistent results for a period of time. These results are not impacted by subsequent changes, such as additions or updates of features and rules.

The configurator snapshot is the collection of the relevant data required for the constraint solve. It includes the variability, such as families, features, models, and the configurator rules.

To optimize the solve time, Teamcenter creates a snapshot of data frozen at a time for a given configuration. This configuration includes the following:

- The configurator context
- The revision rule used
- The effectivity criteria applied
- The rule date to instruct the system to only consider changes that were made on or before this date.

  This implies that changes that were made after this date are not considered.

  It is not possible to consider future changes. Therefore, the rule date can never be a date in the future. **Effectivity** dates can refer to the future dates. The **Effectivity** date is expected to specify a date later than the rule date, but the system does not enforce this expectation. You only need to configure for an effectivity date that precedes the rule date if you are making changes that are expected to take effect prior to saving these changes. For example, this can be done if you are cataloging archeological artifacts.

Typically, this configuration is provided in Product Configurator on Rich Client — Usage in the rule date breadcrumb.

You never interact with the snapshot directly or manage it. However, you can choose the configuration the system uses for the desired action by selecting appropriate values in the configuration definition. This guides the system to choose the most appropriate snapshot, if one exists.

Setting a rule date

When you open a configurator context, a default configuration is applied.

> Note:
>
> A default configuration is based on the revision rule specified in the **Cfg0DefaultRevisionRule** preference.

If the revision rule with which you open a configurator context specifies a **rule date**, Teamcenter finds and uses a valid snapshot for the given rule date. The configurator snapshot used by the system is shown in the rule date breadcrumb.

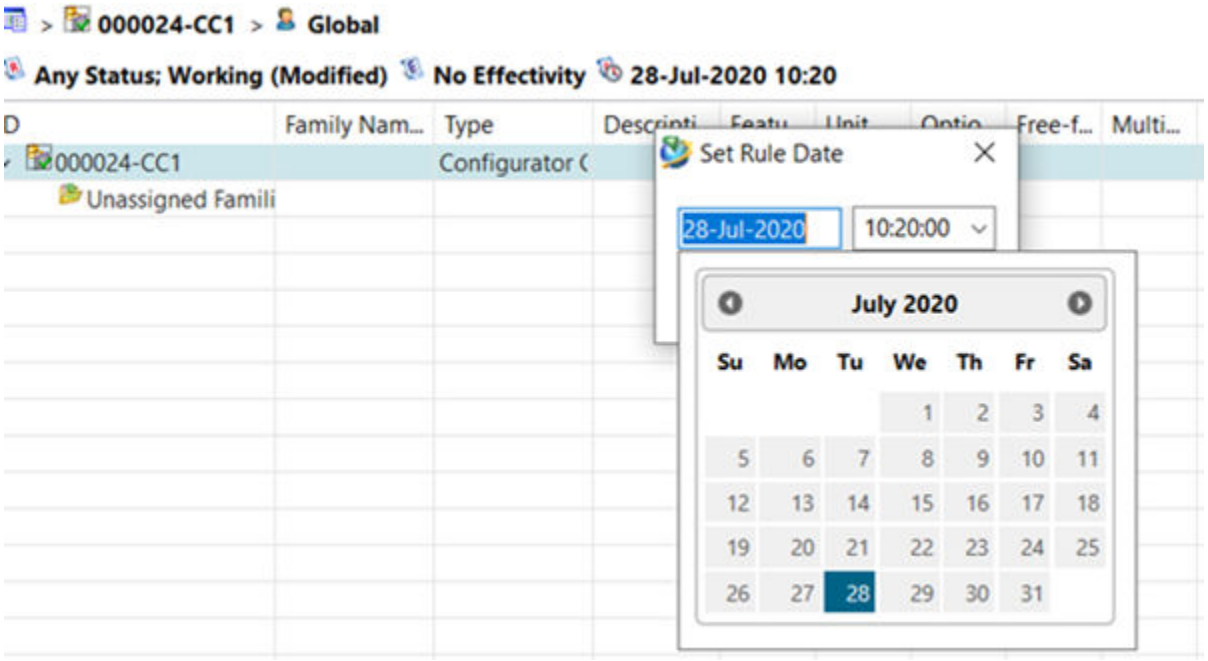
Teamcenter applies the current timestamp rule date with **Most Recent** to indicate that the system displays the latest available snapshot.

This example displays the rule date as the configuration of the selected snapshot based on the system settings set by a Teamcenter administrator. Typically, a Teamcenter administrator ensures that there is a valid snapshot always available in the system for the default configuration. If the system does not find a valid snapshot available, Teamcenter computes a new configurator snapshot.

When you work on a configuration other than the default, you can change the revision rule, effectivity, or the rule date. When changed, Teamcenter finds a valid configurator snapshot for the new configuration. If the valid snapshot is not found, then Teamcenter generates a new configurator snapshot. You cannot use future dates for the rule date.

> Note:
>
> You can set the default rule date by using the **Cfg0DefaultRuleDateForPCA** user preference. By default, the value is **Latest**. For information about retrieving a list of preferences, see *Where can I get a list of preferences?* in *Teamcenter Preferences*.

For example, you can change the rule date in the breadcrumb to **28-Jul-2020 10:20**.



> Note:
>
> You can select time, in seconds, in the **Set Rule Date** window, but the date breadcrumb link does not display the seconds.

Using a snapshot

The configurator objects, such as families, features, and rules, can be modified, released, or revised. An object which is in an unreleased state today may be released tomorrow. When you set rule date **X**, the system searches a snapshot that was compiled for a date **X >= snapshot date and no changes in configuration data of snapshot**. If more than one snapshot is found, the one whose date is closest to **X** is selected. If less than one is found, a new snapshot is created automatically for the requested rule date.

- If there is no change in the configuration data, the system reuses the existing snapshot.
- If there is a change in the configuration data, Teamcenter generates a new snapshot for the same configuration, instead of reusing the existing one.

If there is no change to the configuration data from the last snapshot generated to the current date, the system reuses the existing snapshot. As a result, the system load is reduced and solve response times improve because the number of solve requests that reuse the same snapshot is increased and fewer snapshots have to be created. This provides accurate results over time. The administrator can configure this period to balance the response time and how dated the snapshot is based on your business needs.

The creation time of a snapshot is not the snapshot time. For example, you can create a snapshot for a *milestone* configuration in the past. However, the snapshot creation date is today. The snapshot time is the *milestone* date.

By default, Product Configurator on Rich Client — Usage uses the revision rule specified with the **Cfg0DefaultRevisionRule** preference. If the revision rule has specified a rule date, Teamcenter automatically uses the rule date for this revision rule. The offset calculation takes place when the breadcrumb or Product Configurator objects are initialized with a **Revision Rule** that does not have a rule date, such as a **Revision Rule** specified in the **Cfg0DefaultRevisionRule** preference. The **Cfg0RuleDateOffset** preference specifies the offset, in seconds, to add to last midnight when setting the rule date.

If the offset is 0, the default value for this preference, the rule date is set to today 12:00 AM. A preference value of 21600 causes the rule date to be set to today 06:00 AM, or midnight + 21600 seconds. A preference value of -10800 causes the rule date to be set to yesterday 09:00 PM, or midnight - 10800 seconds. The value of the preference cannot be more than 86400 seconds, or 24 hours. If the calculation of the offset results in a future date or time, the system uses the date as one day earlier, instead of today. The default value of the offset is 0 seconds, or 12:00 AM.

Example: Using the snapshot when the input rule date is changed for the same configuration

For the scenario below, each row is an independent task. The system or server time (GMT) is the time at which you perform a task.

You create the following configuration for your configurator context.

Snapshot validity window	Until a change in data, including models, families, features, allocations, model families, packages, rules, and global rules
Revision rule	Any Status; No Working
Effectivity	Not applied

The following snapshots exist in the system:

Snapshot	Configured As date
1	**21-Apr-2020 02:00:00**
2	**21-Apr-2020 10:00:00**
3	**21-Apr-2020 17:00:00**

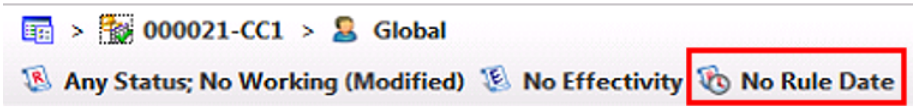
The calculated snapshot search is based on the input rule date and current snapshot date, with no change in the configuration data in the above three snapshots.

Task	System time	Input rule date	System behavior
1	**21-Apr-2020 17:55:00**	**21-Apr-2020 17:55:00**	The snapshot #3 for **21-Apr-2020 17:00:00** falls within the snapshot validity period (no change in data). Hence, it is reused.
2	**21-Apr-2020 17:55:00**	**21-Apr-2020 16:55:00**	The snapshot #2 for **21-Apr-2020 10:00:00** falls within the snapshot validity period (no change in data). Hence, it is reused.
3	**21-Apr-2020 19:00:00**	**21-Apr-2020 10:40:00**	The snapshot #2 for **21-Apr-2020 10:00:00** falls within the snapshot validity period (no change in data). Hence, it is reused.
4	**21-Apr-2020 19:05:00**	**21-Apr-2020 02:30:00**	The snapshot #1 for **21-Apr-2020 02:00:00** falls within the snapshot validity period (no change in data). Hence, it is reused.

Task	System time	Input rule date	System behavior
5	**21-Apr-2020 19:05:00**	**21-Apr-2020 01:30:00**	No snapshot is found. A new snapshot #4 is generated for **21-Apr-2020 01:30:00**.
6	**21-Apr-2020 19:00:00**	**22-Apr-2020 03:00:00**	The input date is in the future. You cannot use a future date, so the system generates an error.

Example: Using the snapshot when no specific rule date is selected

You can use a revision rule which has **No Rule Date** set for the constraint solve.



This means that you are asking the system to select the latest state and process. In this case, where no rule date is selected, the system considers the current system time at which the request is made, and then follows the new change in configuration data.

For the scenario below, each row is an independent task. The system or server time (GMT) is the time at which you perform a task.

You create the following configuration for your configurator context.

Snapshot validity window	Until a change in data, including models, families, features, allocations, model families, packages, rules, and global rules
Revision rule	Any Status; No Working
Effectivity	Not applied

The following snapshots exist in the system:

Snapshot	Configured As date
1	**21-Apr-2020 02:00:00**
2	**21-Apr-2020 10:00:00**
3	**21-Apr-2020 17:00:00**

The calculated snapshot search window is based on the input rule date and current snapshot validity window.

Task	System time	Input rule date	System behavior
1	**21-Apr-2020 16:30:00**	No rule date	The snapshot #2 for **21-Apr-2020 10:00:00** falls within the snapshot validity period (no change in data from **21-Apr-2020 16:30:00** to **21-Apr-2020 10:00:00)**. Hence, it is reused.
2	**21-Apr-2020 23:00:00**	No rule date	The snapshot #3 for **21-Apr-2020 17:00:00** falls within the snapshot validity period (no change in data from **21-Apr-202 23:00:00** to **21-Apr-2020 17:00:00)**. Hence, it is reused.
3	**21-Apr-2020 07:00:00**	No rule date	The snapshot #1 for **21-Apr-2020 02:00:00** falls within the snapshot validity period (no change in data from **21-Apr-2020 07:00:00** to **21-Apr-2020 02:00:00)**. Hence, it is reused.
4	**21-Apr-2020 20:00:00**	No rule date	If there is a change in data (for example, a new family is added), a new snapshot is created for **21-Apr-2020 20:00:00**.

Example: Working with unreleased and modifiable data

To configure a structure for past dates and to access an object’s old state, you must maintain the object's revision history.

Each clause in a revision rule has zero, one, or multiple matching candidate revisions:

- If there are zero candidate revisions, processing skips to the next clause in the revision rule. If there are no more clauses in this revision rule, the object has no configured revision.
- If there is exactly one candidate revision, then this is configured as the configured version, and the revision rule configuration for this object ends. It moves on to configuring the revision for the next object.

- If there are multiple candidates, one of them must be chosen as the configured revision. This choice is based on:
  - **Latest Creation** date for working data.
  - **Latest Release Status** date for clauses that are specific to a given release status.
  - **Latest Release** date of the workspace object for **Any Status** clauses.

For this example, you create the following variability:

Family	Feature
Engine	V4
	V6
Fuel	Petrol
	Diesel

Next, you create the new product model family named **Models** with the following data:

Object/Revision	Creation Date	Last Modified Date
Models/001 (Model Family) • LXI/001	23 November 2016 – 3:54 (T1)	23 Nov 2016 – 3:54 (T1)

> Note:
>
> In this example, T1 represents the creation date and time, which is 23 November 2016 – 3:54.

At T1, LXI does not have any member features.

If you apply the rule date of T1 and the **Any Status; Working** revision rule on the following configuration and expand, you get the following result:

Configuration	Expand result	Reason
LXI	LXI	V4, Petrol are not model members associated with LXI.

Next, you modify the **LXI/001** model and add V4 & Petrol as new members as follows:

Object/Revision	Creation Date	Last Modified Date
LXI/001 • V4 • Petrol	23 November 2016 – 3:54 (T1)	23 November 2016 – 4:00 (T2)

If you apply the rule date of 23 November 2016 – 4:05 (T3), instead of T1 (23 November 2016 – 3:54) or T2 (23 November 2016 – 4:00) and the revision rule **Any Status; Working** to the same configuration, the results are as follows:

Configuration	Expand result	Reason
LXI	LXI &V4 & Petrol	V4, Petrol are now model members associated with LXI.

What happens when you use a rule date that is in the past?

Consider the following scenario in our example: If you apply a rule date of T1 or a date shortly after T1 and the revision rule **Any Status; Working** to the same configuration, the results are as follows:

Configuration	Expand result	Reason
LXI	LXI & V4 & Petrol	LXI/001 is configured because its create time is T1. At T1, the configuration LXI/001 did not have the member V4, Petrol. Although the model members V4 and Petrol were assigned to model LXI with a date later than the rule date, the system assumes that the model was always intended to be associated with these model members and that it is not necessary to go back to verify the system behavior again before saving this change to LXI.

Create a new revision to maintain the object revision history

Configuring WIP data with a rule date that is in the past might bring in some aspects from a future configuration scenario. In our example, the *future* configuration scenario is that a Fuel and Engine value were saved as model members of LXI.

Teamcenter assumes that any change that is saved to working objects is not business relevant with respect to the revision rule configurations for a date that is in the past. It assumes that the user might have only changed the description or some other non-business relevant property. Additionally, it assumes that if a business-relevant change was saved, the earlier object state is not business relevant because it was neither used to configure and build a prototype nor was a product order ever used to configure a product structure in a way that would require the business to restore the same configuration results at a later date. In such cases, you can save the changes to a WIP object. The system performs the configuration considering the new property value, as if it was there right from the beginning when the rule was created.

If, on the other hand, the earlier configuration must be restored because a price was quoted for it or a prototype was built for testing, then it is recommended that the object's state be preserved by *releasing* it. You can then save any updates on a newer (WIP) revision of the object.

To configure the LXI model, you must create a new *revision* of the model when you add or modify its membership or other properties.

Object/Revision	Creation Date	Last Modified Date
Models/001 (Model Family)		
LXI/001 (no members)	23 Nov 2016 – 3:54 (**T1**)	23 Nov 2016 – 3:54 (**T1**)
LXI/002 with two members: • V4 • Petrol	23 Nov 2016 – 4:00 (**T2**)	23 Nov 2016 – 4:04 (**T4**)

> Note:
>
> In this example, T4 represents the date and time, which is 23 November 2016 – 4:04.

If you apply the rule date of T3 (23 November 2016 – 4:05) and revision rule **Any Status; Working** to the following configuration, the results are as follows:

Configuration	Expand result	Reason
LXI	LXI & V4 & Petrol	The system configures the latest working version, LXI/002. V4 and Petrol are added as model members associated with LXI/002.

If you apply the rule date of T1 (23 Nov 2016 – 3:54) and revision rule **Any Status; Working** to the same configuration, the results are as follows:

Configuration	Expand result	Reason
LXI	LXI	The system configures LXI/001 because LXI/002 is created at T2 (23 Nov 2016 – 4:00). This is later than the rule date. V4 and Petrol are not model members associated with LXI/001.

Releasing configurator data

You can use the out of the box Enterprise change process to release, status and revise configurator objects. Individual Product Configurator on Rich Client — Usage features such as features and families, configurator rules, configurator allocation, and partitions may be the target of a workflow, for example, a review process workflow.

Your administrator can configure workflows to use specific workflow handlers to ensure that when you release configurator features, you can also automatically include their families and configurator allocations in the same workflow process.

For example, your administrator can configure a workflow to initially look for WIP revisions to attach them as targets, and then looks again for released revisions to attach them as reference.

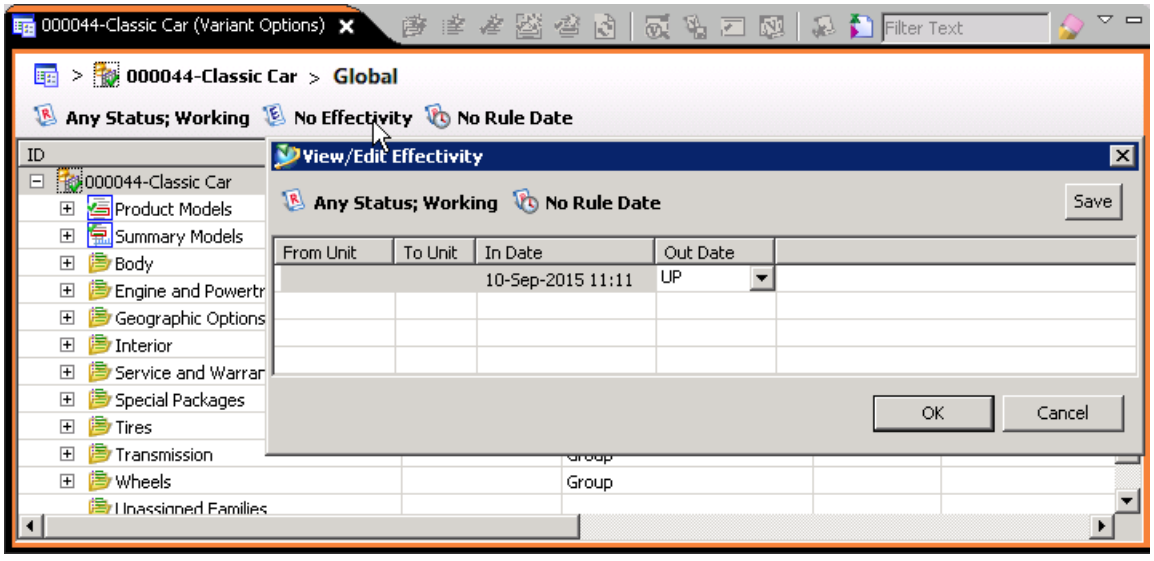
You can discontinue a revision of a configurator object, such as a family, model family, feature, configurator allocation, model, and rule.

Applying effectivity on your configurator data

In Product Configurator on Rich Client — Usage, configurator features can be revised and can have effectivity. Effectivity is supported on models, all types of features, including the feature allocation. You can also apply effectivity on all configurator rules. Currently, effectivity is not supported on family, groups, and their allocations.

Object type	Revise	Status	Effectivity
Feature and feature allocation	Yes	Yes	Yes
Family and family allocation	Yes	Yes	Yes
Group and group allocation	No	Yes	NA
Configurator rules	Yes	Yes	Yes
Variant rules	NA	Yes	NA
Variant criteria	Yes	Yes	NA

Effectivity allows you to control when each feature is introduced and obsoleted. Effectivity also allows you to configure configurator contexts according to specified intents, such as unit numbers, date conditions, or a combination of features. You must specify effectivity directly, for example, `Effective September 10 2015`.



Note:

The effectivity model in Structure Manager is different from the effectivity model in Product Configurator and 4th Generation Design objects. Even though the representation in the user

interface may seem the same, the underlying data model, processing and mechanisms are distinct.

You can configure open-ended effectivity with an infinite end date or unit number (**UP**) or until stock is out (**SO**). For example, **1–May-2015 to UP** or **1–May-2015 to SO**. **UP** and **SO** are both provided to aid integration with ERP systems but, in Teamcenter, they are functionally the same.

Note:

Your administrator can set the **PCA_effectivity_shown_columns** preference to **All** to show both date and unit effectivity in the **Effectivity** view.

All types of features, such as literal features, compound feature summaries or feature packages, summary models and models, can have an individual effectivity expression assigned.

The features are configured based on the effectivity expression set on the revision rule. The following example illustrates the configuration result based on the effectivity of a feature and the revision rule being applied.

#	Feature effectivity	Revision Rule effectivity	Configuration result
1	**1–June-2015**	**1–April-2015**	Not configured
2	**1–January-2015 to 30–September-2015**	**1–April-2015**	Configured
3	**1–May-2015 to UP**	**1–April-2015**	Not configured
4	**1–June-2015 to UP**	**1–October-2015**	Configured
5	**1–January-2015 to 30–September-2015**	**1–October-2015**	Not configured
6	**1–May-2015 to UP**	**1–October-2015**	Configured

You can deactivate a feature by setting an *effective out date* corresponding to when it should be deactivated. If no features in a family are effective, then the family generates an order expand error if the family is required. Deactivating a feature prevents it from being allocated to new product contexts. You may not make a configurator allocation apply retroactively, such as, if the feature is not valid at the date the allocation action is taking place or at some date in the future, it may not be allocated. In the case the feature is valid only in the future, the allocation may not apply until that future date.

You can reactivate a deactivated feature by setting an *effective in date*.

Note:

You can discontinue all configurator objects, including a family, model family, feature, configurator allocation, model, and rule. A discontinued revision allows you to capture business data using object properties on the discontinued revision. Additionally, you can discontinue revisions of configurator objects for a sub range of effectivity.

You can configure configurator view for the following:

- Show data that is effective for a selected date or unit, such as a date other than today.
- If you are using effectivity criteria relative to future dates, configure data as of now to show the currently planned data effective for a future date.
- Specify an independent effective-for date or date ranges.

  These date ranges typically include dates that are later than the as-of date. However, you can specify an effective-for date that precedes the as of date, for example, if you notice an error in your effective-for conditions for past dates and you want to correct it. In this example, you can compare the data effective for a specified past date as it existed prior to your correction and now.

  If you specify an effectivity date range in a revision rule, Teamcenter finds the configurator object revisions according to their released status and released date. These configurator objects may have partially overlapping, discontinued, or cutback effectivity ranges. A precedence filter selects the most applicable revision for the given effective point.

> Tip:
>
> To set the effectivity to a single unit (for example, unit 12), specify the same value for the effect-from and effect-to points. In this example, you set the **From** value in the **Effectivity view** to 12 and leave the **To** value blank. Using the default solve, Teamcenter never effects out unit 12.

Configurator rules are applied only if an input effectivity is precise, such as **Unit = 1**, or **1–April-2016**, or if there is no input effectivity. However, configurator rules are validated irrespective of a range or precise effectivity.

In this scenario, **M1** is a feature in the family **M**. **A1** and **A2** are features in the family **A**. The inclusion rule is written as **M1→A1**, which means that **M1** includes **A1**. The inclusion rule effectivity is set to **Unit = 1–20**. When you select **M1**, then **A1** is included in the configuration when an input effectivity is set to **Unit = 1–20**.

User selection	Configuration Result	Meaning
M1 with input effectivity **Unit = 1–UP**	**M1**	Rule is not applied because an input effectivity is a range.
M1 with input effectivity **Unit = 5**	**M1** and **A1**	Rule is applied because an input effectivity is precise.

The following shows the results of validation, such as when you make your selections, apply effectivity, and validate your selections.

User selection	Configuration Result	Meaning
M1 and **A2** with input effectivity **Unit = 5**	**Invalid**	An inclusion rule states that **M1** includes **A1** within an effectivity of **Unit = 1–20**.
M1 and **A2** with input effectivity **Unit = 5–15**	**Invalid**	An inclusion rule states that **M1** includes **A1** within an effectivity of **Unit = 1–20**.
M1 and **A2** with input effectivity **Unit = 25–35**	**Valid**	Inclusion rule no longer applies because an input effectivity is now set to **Unit = 25–35**.

Example: Using precedence filtering when configuring data using effectivity ranges and revision rules

1. The **Body Type** family is created with the following features and effectivity ranges. Earlier revisions of those features are released, revised, and have different effectivity ranges.

   The following revision rule and effectivity range is applied.

Effectivity Range	Revision rule
Unit=1...35	Any Status;Working

   The following configuration is displayed.

Object/Revision	Date Released	Effectivity
Body Type/001		
Coupe/001	1-Jan-2015	Unit=1...10
Coupe/002	1-Apr-2015	Unit=8...18
Coupe/003	1-Jun-2015	Unit=14...28
Coupe/004		Unit=11...35

2. The following revision rule and effectivity range is applied.

Effectivity Range	Revision rule
Unit=16...30	Any Status;Working

   The following configuration is displayed.

Object	Date Released	Effectivity
Body Type/001		
Coupe/003	1-Jun-2015	Unit=14...28
Coupe/004		Unit=11...35

The following steps show how Teamcenter arrived at these results using precedence filtering:

- Applying the **Any Status;Working** revision rule assigns the following precedence ranking for the feature revisions that partially or fully satisfy the effectivity range.

Feature/revision	Ranking	Release date
Coupe/003	0	1-Jun-2015
Coupe/002	0	1-Apr-2015
Coupe/004	1	

  The ranking set to 0 represents the **Any Status** clause in the revision rule and the ranking set to 1 represents the **Working** status. The **Coupe/003** feature revision is ranked before the **Coupe/002** feature revision because **Coupe/003** has a newer release date.

- Teamcenter evaluates each feature revision in the ranking list order to determine the configuration. Once a partial or full effectivity range match is found, that feature revision is displayed. If a partial match was found, Teamcenter then identifies the next preferred revision of that feature as per precedence criteria that satisfy the *remaining* effectivity ranges. The process of identifying the preferred revisions is repeated until the entire effectivity range criteria is satisfied by selected revisions or there are no feature revisions left.

  a. The **Coupe/003** feature revision is displayed because the **Unit=14...28** effectivity range partially satisfies the **Unit=16...30** effectivity range. The partial match is the **Unit=16...28** effectivity range. The reduced remaining effectivity that needs to be satisfied is now the **Unit=29...30** effectivity range.

  b. The **Coupe/002** feature revision overlaps the partial **Unit=16...28** effectivity range that is already satisfied by the **Coupe/003** feature revision. The **Coupe/002** feature revision is skipped and not displayed.

  c. The **Coupe/004** feature revision satisfies the remaining **29...30** effectivity range and is displayed.

Validating effectivity

Validation based on an effective in date

An *effective in date* of a configuration object cannot be older than the creation date of that object on which an effectivity is set. If this validation fails, Teamcenter generates an error and effectivity is not saved. This validation is applicable to all supported configurator objects.

> Note:
>
> An object cannot be effective before its creation date.

Effectivity validation with regards to other revisions

Effectivity can be validated with regards to other revisions, including all revisions, configurator allocations and member revisions of a configurator business object, if any. If this validation fails, Teamcenter generates a warning and you have an option to continue. There are different validation criteria for each configuration object.

Configurator business object	Validation Criteria
Feature allocation	A union of effectivity of all feature revisions referring to this allocation must include effectivity of this configurator allocation.
Feature package allocation	• A union of effectivity of all feature revisions referring to this configurator allocation must include effectivity of this allocation. • Effectivity of feature package allocation has to be narrower than the effectivity of its members' allocation of same product.
Feature summary allocation	• A union of effectivity of all feature revisions referring to this configurator allocation must include effectivity of this allocation. • Feature summary allocation effectivity must satisfy union of effectivity of all member allocation revisions of same product.
Feature	No validation.
Feature package	Feature package effectivity has to be narrower than the effectivity of all of its members.
Feature summary	Feature summary effectivity must satisfy a union of effectivity of all member revisions.
Model	• Model effectivity has to be narrower than the effectivity of all of its member revisions. • Model effectivity has to be wider than available features effective date.

Configurator business object	Validation Criteria
Summary model	Summary model effectivity must satisfy a union of effectivity of all member revisions.
Product line	Product line effectivity has to be narrower than the effectivity of all of its member revisions.

Example of effectivity validation with regards to other revisions

You create configurator data with the following values.

Family	Features	Effectivity	Type
Color	**Red**	**Unit=1–100**	Feature
	Blue	**Unit=1–50**	Feature
	Green	**Unit=1–100**	Feature
Engine	**V4**	**Unit=1–30**	Feature
	V6		Feature
	V8	**Unit=1–80**	Feature
Models	**LXI**	**Unit=1–50**	Product model
	VXI	**Unit=1–60**	Product model
	ZXI		Product model
Package family	**ColorAndEngPkg**		Package family
Summary family	**Color Summary**		Summary family
Summary model family	**Summary Models**		Summary model family

You add the following feature to models, package, summary, and summary model family.

Family	Features		Effectivity
Models	**ZXI**	**Green**	**Unit=1–100**
		V8	**Unit=1–80**
Package family	**ColorAndEngPkg**	**Blue**	**Unit=1–50**
		V4	**Unit=1–30**
Summary family	**Color Summary**	**Red**	**Unit=1–100**
		Blue	**Unit=1–50**
Summary model family	**Summary Models**	**LXI**	**Unit=1–50**
		VXI	**Unit=1–60**

The validation results are as follows.

Configurator objects	Input effectivity	Result
ColorAndEngPkg	**Unit=1–30**	No warning
	Unit=1–50	Warning Not all members are effective.
Color Summary	**Unit=1–100**	No warning
	Unit=51–100	No warning
	Unit=101–200	Warning Not all members are effective.
	Unit=100–200	No warning
Summary Models	**Unit=51–60**	No warning
	Unit=61–100	Warning Not all members are effective.
	Unit=1–100	No warning
Model **ZXI**	**Unit=1–85**	Warning Not all members are effective
	Unit=1–60	No warning
Configurator allocation of color **Red** in **VXI**	**Unit=1–105**	Warning Effectivity of an allocation must be narrower than its feature revisions.
	Unit=1–100	No warning

Note:

Effectivity validation on configurator allocation of a feature package and feature summary is similar to:

- Feature allocation
- Feature package or feature summary, respectively

Validating effectivity with regards to effectivity of related objects

An input effectivity is validated to ensure the effectivity is consistent with effectivity of related configured-in objects. This validation considers the applicable *revision rule and effectivity*, if an effectivity is set on the revision rule.

Configurator business object	Related cross objects considered in the validation
Feature	Rules, package, summary model family, summary model
Feature package	Rules, model, package members
Feature summary	Rules, summary members
Product line	Rules, summary model, package
Product model	Rules, summary model, package
Summary model	Rules
Global rule	Other global rules
Rule	Rules, features, package, summary

For example, you have a feature **A1** configured with **Any Status; Working** revision rule and an effectivity of the revision rule set as **Unit=1–100**. If you set a disjoint effectivity, such as **Unit=1–100**, you see a validation error.

Configurator objects not considered in effectivity validation

Effectivity validations on family, family allocation, group, group allocation, and admissibility are not supported.

Example of effectivity validation with regards to other objects

When you update membership in either feature package, feature summary, product model, product line and summary model, the assigned effectivity is validated.

For example, your **Standard package** is effective for **Unit=1–10**. You decide to add a new feature to that package, for example, **Silver Trim** which is effective for **Unit=11–20**. The system generates a violation error message.

You also see a validation error, if any of the related objects have disjoint effectivity with an object under validation.

Family	Features	Effectivity	Type
Color	**Red**	**Unit=1–100**	Feature
	Blue	**Unit=1–50**	Feature
Summary family	**Color Summary**	**Unit=1–50**	Package family

You add the following features to your summary family.

Family	Features		Effectivity
Summary family	**Color Summary**	**Red**	**Unit=1–100**
		Blue	**Unit=1–50**

If you change a feature summary effectivity for **Color Summary** to **Unit=51–100**, the system generates a validation error.

Example of effectivity validation using the data configurator snapshot

Teamcenter retains configuration snapshots of variant data for configurable objects to prevent repeated frequent database queries for the same set of configurable objects. This improves the performance during the validations and allows you to continue to work with your data, without any concurrent updates or interruptions during the specified interval of time. However, the decrease in the interval time results in slower performance.

> Note:
>
> The configurator snapshot is created only when there is a change in the configurator data, for example, when variability data is added or modified or when configurator rules are created or updated. If there is no change, then the system points to the latest snapshot based on the rule date selected in the UI.

In this scenario, the current system time is set to **11-August-2020 9:45**.

You create configurator data with the following values.

Configurator Object	Features
Family **A**	**A1**
	A2
Family **B**	**B1**
	B2
Package family **P1**	Contains: • **A1** • **B1**

You create the following inclusion rule:

- **A1** includes **B1**

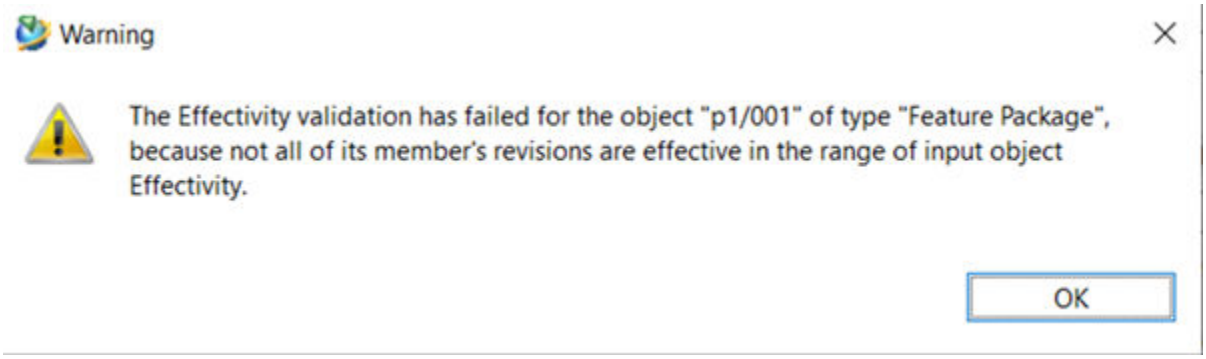
Once the data is saved, a configurator snapshot is be created with timestamp **11-August-2020 9:45**.

Now, the current system time is set to **11-August-2020 10:10**.

Assign effectivity as follows.

Configurator Object	Features	Effectivity
Family **A**	**A1**	**Unit=1–5**
	A2	
Family **B**	**B1**	**Unit=1–10**
Package family **P1**	Contains: • **A1** • **B1**	**Unit=11-20**

The system generates a warning message for the package family **P1** because not all members are effective for the selected effectivity.



The data is validated against other revisions without considering configurator rules. Such validation is done by using the data directly from the database and not by using the configurator snapshot.

Example of effectivity validation with the same creation date and input effectivity

You create configurator data with the following values.

Family	Features	Type
Color	**Blue**	Feature
	Green	Feature

On **29 September 2016 1:58 pm**, you create two default rules.

- **Color** = **blue**

- **Color = green**

On **29 September 2016 2:00 pm**, you assign effectivity to both default rules.

Default rule	Effectivity Start Date	Effectivity End Date
Color = blue	**29 September 2016**	**30 September 2016**
Color = green	**1 October 2016**	**2 October 2016**

The system stores the effectivity as follows:

Default rule	System Assigned Effectivity Start Date	System Assigned Effectivity End Date
Color = blue	**29 September 2016 2:00 pm** An object cannot be effective before its creation. Thus, this default rule is not effective from midnight of **29 September 2016.**	**29 September 2016 11:59:59 pm**
Color = green	**1 October 2016 12:00 am** Because an object was created before **1 October 2016** , the default rule is effective from midnight of **1 October 2016** .	**1 October 2016 11:59:59 pm**

If you apply defaults with the effectivity range from **29 September 2016** to **30 September 2016**, the system applies defaults as **29 September 2016 12:00am** to **29 September 2016 11:59pm**.

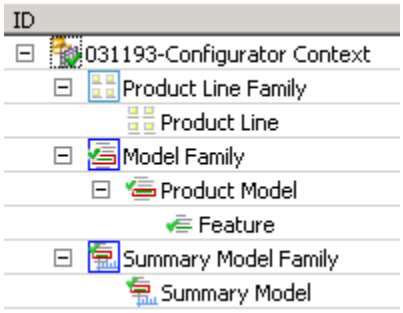
Because default rules are not effective prior to **29 September 2016 2:00pm**, no color is set as a default during validation. The **Color = blue** default rule is applicable for a partial effectivity range. An expansion of configuration can only take place if only one feature from a selected family is available for a complete input configuration. Therefore, the system is not returning any default for the specified range of **29 September 2016 12:00am** to **29 September 2016 11:59pm**.

If you change an input effectivity range to **1 October 2016** to **2 October 2016**, then the **Color = green** default rule is effective and applied during configuration.

Defining models

Product models

A product model represents a specific market-facing product to be offered. Specific features from a product line can be made available for each product model.



Summary models

A summary model is a collection of models that allows you to summarize product models in a configurator context. Summary models used in rules imply an OR combination of all product models that they summarize. A summary model behaves similarly to a feature summary. In any configurator rule, you can refer to the specific discrete product models or to the summary model.

Summary model groups models that appear in different branches of the product line hierarchy. This grouping is based on common technical characteristics, such as common size.

For example, you can collect all models with a hatchback body style into a summary model family and then create rules that apply to all models within that summarized set. When creating rules, you can now choose the summary model. You do not have to list every single model with the hatchback body style. If you add a new model with the hatchback body style to that summary model, all existing rules that apply to all hatchback styles also automatically apply to the newly added model.

Package models

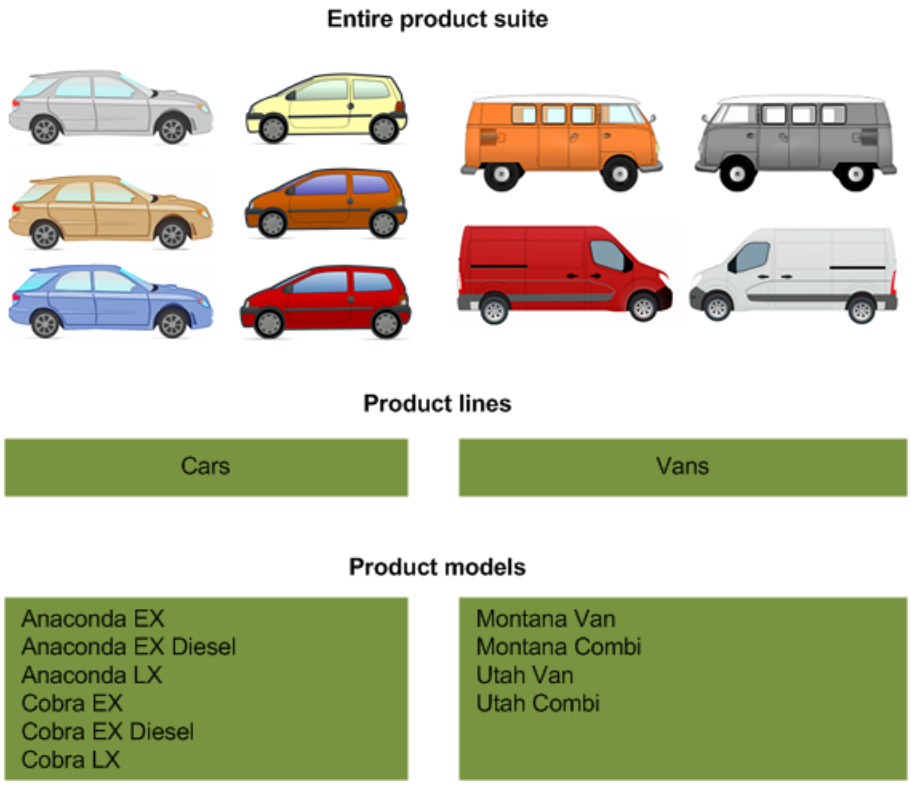
Typically, package models are used if there are specific key features that are always included for the model, and thus they help to define the model.

> Note:
>
> To create a product model, product model family, summary model, or summary model family, you must have an appropriate license level. If the license is not available, the system generates an error during the creation of the respective object.

Defining product lines

Product lines

Product lines structure product models into a hierarchical tree based on common business characteristics. The breakdown is driven by financial, legal, geographic, and marketing aspects. This allows you to create and manage rules for these related products. The breakdown of product models into product lines can drive your company's business processes outside configurator such as development using programs and setting of common targets.



Cars and vans can be treated as two different product lines that can be represented with two configurator contexts. Typically, product managers determine what product lines are offered and what features are introduced or carried over for each. Product models are market-facing product variations within a product line. At this level, there is more specific refinement of exactly which product features are offered per product model. Engineering and product design is typically completed and certified for one or more product models.

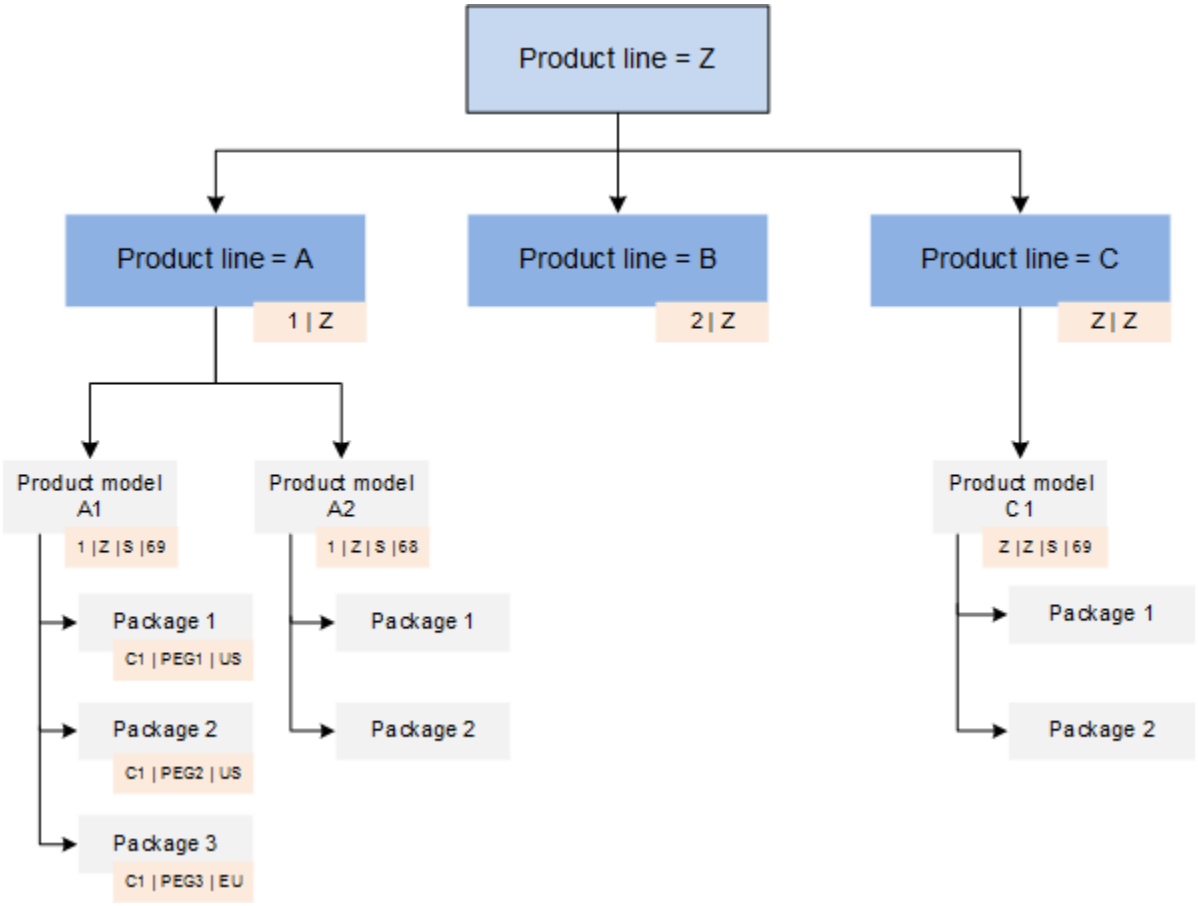
Product line is a group of products that are closely related to each other by function, customer group, market, or price range, etc. It is less detailed and less precise than product models as multiple models can be a part of a single product line.

For example, companies offer a suite of products to the market. These are often broken down by type of product or product line. Within a product line, there are typically one or more market facing product models that can be similar but can offer different combinations of features. For each product line and each product model within that configurator context, you can designate a relevant subset of features.

The difference between a product line and a configurator context

Product lines are used to group related product models. The grouping of product models or product lines is achieved by using model membership.

The example below displays product lines for a car.



A *configurator context* represents a family of products that may share the same set of groups, families, and features. Business practices determine the reuse of a configurator context between product lines. If there is a significant reuse between product lines, then they can use the same configurator context for multiple product lines. On the other hand, all product models from the same product line share the same configurator context as the product line.

Discontinuing revisions of configurator objects

You can discontinue all configurator objects, including a family, model family, feature, configurator allocation, model, and rule. A discontinued revision allows you to capture business data using object properties on the discontinued revision. For example, based on your business process, you set specific properties on the discontinued revision or occurrence that communicates information to downstream applications.

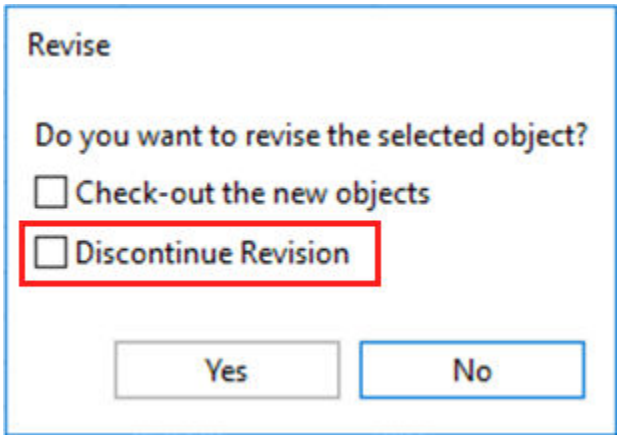
The advantage of discontinuing a configurator object over using an explicit cutback is that explicit cutback can only be applied after the change is approved. Discontinuation is applied while the change is in progress, and it does not impact the released configuration until the change is released.

You can discontinue a released revision of the configurator object from the configurator context. Additionally, you can discontinue revisions of configurator objects for a sub range of effectivity.

For example, you may have the following revisions:

Rev1/01 Released (1 -20)
Rev1/02 Released & Discontinued (20-30)
Rev1/03 Working & Continue (30 –UP)

Discontinue a revision by choosing **File** → **Revise** and then selecting **Discontinue Revision**.



The system shows no revisions if the revision rule configures a discontinued revision. You can locate a discontinued revision in My Teamcenter. You cannot revise it. If the latest revision of an object was discontinued, you can locate a previous revision and then revise it.

When you discontinue an object in **Global** mode, the system does not automatically discontinue it in **Context** mode.

For example, if you discontinue Revision 2 of the family or feature in **Context** mode, the system does not allow you to create Revision 2 of that family or feature in **Global** mode.

> Note:
>
> In the current release, Product Configurator does not visually distinguish discontinued configurator objects.

Impact analysis for configurator objects

You can determine where configurator objects are used by performing an Impact analysis. For example, right–click the feature, or its member, and choose **Open with→Impact Analysis**. The standard Teamcenter Relation Browser displays all the places this feature is referenced in product models, feature summaries and feature packages, configurator rules and saved configurations.

You can perform a where-used query on a feature to find the following:

- All configurator contexts where that feature is allocated to.

- A family and group for the feature.
- Feature summary and feature package.
- All rules that use the configurator contexts with the selected feature.
- Saved configurations.

> Note:
>
> The **Where Used** query returns the list of objects where they have been used. However, the filtering of these objects by **Rule** (when it is a configuration), **Display**, and **Filter By** is not supported for configurator objects.

Defining the solve type for variant configuration criteria

You can define criteria to filter product data based on variant configuration criteria to use the following solve types:

Type	Value	Result
MISMATCH	1	Select objects with variant conditions that do not match the solve criteria. It is usually combined with INVERT.
EXPLICIT	2	Select only objects with variant conditions that explicitly satisfy the solve criteria (positive solve).
COPRIME[4]	4	Select objects potentially satisfying the solve criteria.
TRUE	8	Select objects with a variant condition equivalent to the Boolean constant TRUE.
FALSE	16	Select objects with a variant condition equivalent to the Boolean constant FALSE.
CONDITION	32	Select objects with a nonconstant variant condition.
ERRORCHECK	64	Select objects with variant conditions returning an error when solved.
NOCONDITION	128	Select objects with configurable behavior having no variant condition.

4 Two integral numbers are coprime if their greatest common divisor is 1. Two Boolean expressions are coprime if their greatest common Boolean divisor is TRUE.

Type	Value	Result
NOCONFIGBEHAVIOR	256	Select objects without configurable behavior.
INVERT	512	Invert the filter results, that is, remove what otherwise passes, and pass what would otherwise have been taken out.

Configuring structures by using Product Configurator variants

Advantages of using Product Configurator variants

In classic or modular variants, the variability data such as families, features, and rules are associated with the structures. However, the task to manage the variability data is independent of the task of managing structures. Therefore, it is advisable to separate out the variability data from the structure. Doing so ensures that any changes to the variability data do not change the structure. The Product Configurator application provides dictionaries and configurator contexts to manage the variability data separately. Therefore, different domains, such as system engineering, design, content management, and manufacturing, can use the same variability data without updating a structure.

You can use the Product Configurator variants once the administrator:

- Installs the solution template to support Product Configurator for Structure Manager.
- Sets the preference to enable the Product Configurator variant mode.

Working with variants in 4GD

Associating a configurator context with a 4GD product design

Special considerations apply if you are defining variants to use in a 4GD environment.

You associate a configurator context with a 4GD product design by attaching it to the product design with the GRM relation type specified in the **TC_variant_configurator_relationship** preference. The default value of this preference is **MdI0HasConfiguratorContext**. The product design is the primary object in this relation, and the configurator item is the secondary object.

Alternatively, you can associate item revisions that reference legacy variant data with the product design using the same relationship.

Rolling down variant conditions

Variant conditions roll down from 4GD design control elements to design features, and also from design components to their subordinate design components, in the same way as effectivity conditions. Variant conditions are stored in two properties:

- **mdl0allowed_var_formula**

  Contains the variant condition that the user explicitly assigned to this element.

- **mdl0variant_formula**

  Contains the computed variant condition that takes roll down into consideration.

The computed variant condition in the **mdl0variant_formula** property is always owned locally. It is never exported and always computed from locally available roll down contexts, even if the corresponding 4GD object is a remote replica. Saving a variant condition for a locally owned design control element that controls a remote replica design feature causes Teamcenter to compute a new **mdl0variant_formula** property for the remote replica design feature. Conversely, the **mdl0allowed_var_formula** property of a remote replica design feature is updated only as a result of a Multi-Site Collaboration synchronization.

Writing variant expressions in Teamcenter Normal Form (TNF)

You build variant expressions in Teamcenter Normal Form (TNF) and the Product Configurator displays them in **Variant Expression Editor** grid, as shown in the following examples. They must adhere to the following rules:

- A variant expression consists of one or more subexpressions. Each subexpression is joined by an OR.
- A subexpression consists of one or more expression groups. Each group is joined by an AND.
- An expression group consists of one or more terms. Each term is joined by an OR.
- A term can represent one of the following:
  - A discrete selection in the form of [*namespace*]*family* <*operator*> value, where *operator* can be one of = or !=.
  - One or two range expressions. Each range expression is joined by an AND.
  - A range expression is of the form <*operator*><*valueText*>, where *operator* can be one of >, >=, <, <=.

When you build multilevel expressions:

- The outermost level of an expression represents a separate column in the grid.
- The next level represents selections in different families and selections within the same family if they are not mutually exclusive.

- The next level represents selections within the same mutually exclusive family. This is the family level where only values belonging to the same family are nested, with the exception of non-mutually exclusive selections.
- The next level represents ranges and equalities.
- The innermost level is the literal, indicating the selection value itself.

> Note:
>
> Use a forward slash character `/` to represent `AND/OR`, that is, `EITHER ONE, OR BOTH`. Use a vertical bar character `OR` to represent `ONE OF`.
>
> Binary operators require two operands.
>
> AND and OR are commutative. Teamcenter may not necessarily return the expression in the same order you entered it, providing the expression remains logically correct. For example, you enter `Color=Red AND Engine=V8 AND Accessories=Bells`; Teamcenter may return `Engine=V8 AND Accessories=Bells AND Color=Red`.

The following examples show how common TNF expressions are represented in the grid:

- TNF expression:

  ```
  Color = Red
  ```

  Grid representation:

Color	**Red**	✔

- TNF expression:

  ```
  Width >= 100 AND Width <= 200
  ```

  Grid representation:

Width	**>=100 AND <=200**	✔

- TNF expression:

  ```
  Width <= 100
  ```

  Grid representation:

Width	**<=100**	✔

- TNF expression:

```
GPS != GPS
```

> Note:
>
> GPS is a logical type.

Grid representation:

GPS		🚫

- TNF expression:

```
(Model = 0PC08 OR Model = 0PC35 ) AND
 (Length = 2000 OR (Length = 3000)
```

Grid representation:

Model	**OPC08**	✔
	OPC35	✔
Length	**2000**	✔
	3000	✔

- TNF expression:

```
Width >= 1000 AND Width <= 1500 AND
 ( Length >= 2000 AND Length <= 2500 ) OR
 ( Length >= 3000 AND Length <= 3500)
```

Grid representation:

Width	**>=1000 AND <=1500**	✔
Length	**>=2000 AND <=2500**	✔
	>=3000 AND <=3500	✔

- TNF expression:

```
( ( Model = 0PC08 OR Model = XYZ) AND
( Color != Red ) ) OR ( ( Color = Red ) AND
```

```
( Width >= 1000 AND Width <= 1500 ) AND NOT
( Length >= 2000 AND Length <= 2500 ) )
```

Grid representation:

Model	**OPC08**	✔
	XYZ	✔
Color	**Red**	⊘
Width	**>=1000 AND <=1500**	✔
Length	**>=2000 AND <=2500**	⊘

The TNF expression is a normalized form of the input expression.

- If the expression cannot be satisfied, it always resolves to false and you get an empty TNF vector. For example, `'Engine = Diesel' AND 'Engine = Petrol' and ‘Engine’` is not a multiple selection family.
- If the expression can always be satisfied, it always resolves to true and you get a TNF vector with only one element (a null variant expression). For example, `'ACRequired = ACRequired OR ACRequired != ACRequired'`.

5. Introduce and define variability

Create a configurator context or dictionary

You can create a dictionary in which you can store groups, families, and features. The dictionary provides a convenient method for collecting all of the variant data related to your product suite.

You can also create a configurator context which represents a family of products that may share the same set of groups, families, and features.

You can allocate groups, families, and features directly from a dictionary or from configurator contexts.

> Note:
>
> To create a configurator context or dictionary, you must have an appropriate access and a license level.

1. Click **Product Configurator** in the navigation pane.

   You can also start Product Configurator by right-clicking a configurator context or a dictionary item, group, feature, configurator rule, or saved variant rule in My Teamcenter, and then choosing **Send To → Product Configurator**.

   Teamcenter opens Product Configurator and displays the selected object in the appropriate view.

2. Create a configurator context or dictionary by choosing **File→New→Configurator Context** or **File→New→Dictionary**, respectively.

   Teamcenter displays the **New Business Object** dialog box.

   > Note:
   >
   > Your administrator can create your company-specific custom properties using Business Modeler IDE.

3. Click **Assign** to automatically fill in **ID**.

   > Note:
   >
   > If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas.

4. Click **Assign** to automatically fill in **Revision**.

5. Type a name and description of the configurator context or dictionary.

6. For configurator context only, specify an **availability bias**. Choose from positive or negative.

   You must have an appropriate license level to create a negatively biased configurator context. If the license is not available, the system generates an error during the creation of the negatively biased configurator context.

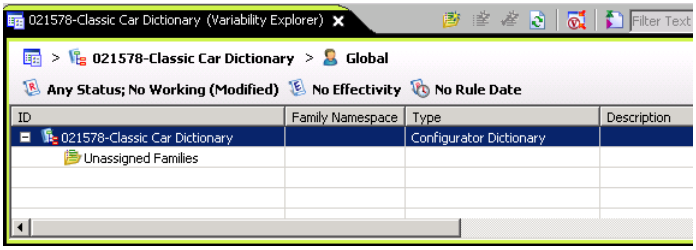
   > Note:
   >
   > You cannot change the availability bias after the configurator context is created.

7. (Optional) Click the corresponding links below to add **Additional Item Information** and **Item Revision Information**.

8. (Optional) Select the **Open On Create** check box to open the newly created object after it is created.

9. Click **Finish**.

   Teamcenter creates the new configurator context or dictionary.



   The **Unassigned Families** group is always included. During allocation, orphan families, such as allocated families not assigned to a group, appear under the **Unassigned Families** group.

> Note:
>
> You can open a Teamcenter object of the **Item** type in Product Configurator and add variability to it. The **Item** type object acts as a configurator context. You can associate families, features, and author and validate configurator rules in the context of this **Item**.

Create groups and families

You can create your variant groups, families, and features for a configurator context or dictionary. Variant data can be reused in multiple product structures and application models (product designs or partition templates).

Caution:

After you create and save a group, a family, or a feature, you cannot delete that object in Product Configurator. You can use the **Cut** command to remove that object from a configurator context or a dictionary.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas. You can then rename your configurator objects, but their IDs remain unchanged.

1. If necessary, create a configurator context or a dictionary by choosing **File→New→Configurator Context** or **File→New→Dictionary**, respectively.

   Teamcenter displays the **New Business Object** wizard, allowing you to create the necessary configurator context or dictionary.

   Alternatively, you can search for and open an existing configurator context or dictionary.

2. Teamcenter opens the configurator context or dictionary in the **Variability Explorer** view, with any existing variant data shown. Any existing variant groups, families, and their features are displayed in the tree table.

   You can send more than one item revision to Product Configurator and each one opens a new **Variability Explorer** view.

3. Select the configurator context or dictionary and click **Add Group** in the view toolbar.

   Teamcenter adds an empty row for the new group.

4. Click in the **ID** field of the empty row and type the ID of the group. The chosen ID must be unique for the current configurator context.

   Teamcenter adds the ID in italics, indicating the entry is not yet saved. An asterisk (*) next to the name of the view also indicates there is unsaved data.

5. Click in the **Description** field of the new row and type the description of the group. Then, press Enter or click the next row.

ID	Family Namespace	Type	Description
000047-Classic Car		Configurator Context	
Transmission		Group	*Classic Car Transmissions*
Wheels		Group	*Wheel Options*
Unassigned Families			
Body			*Classic Car Body Options*

6. Click **Save** to save the new group.

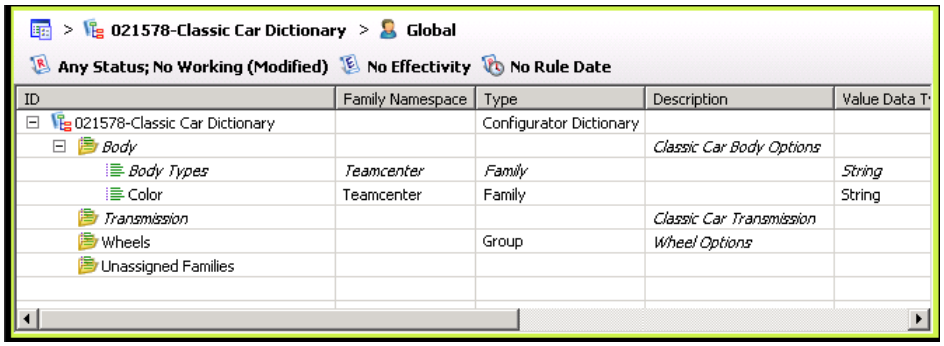
Teamcenter displays the name and description in nonitalic text, indicating the group is saved.

7. Define variant data for the new group by doing one or more of the following:

   - Add a sibling to the new group by right-clicking the new group, choosing **Add Group**, and following steps 3 through 7 previously.

   - Click **Add Family** to add a new family to the selected group. A new empty row appears as the last child of the selected group, and the group is expanded if necessary. When you type the ID of the family, the chosen ID must be unique for the current configurator context or dictionary.

     You can create multiple objects in a configurator context or dictionary. They are shown in italics until you save your changes.



   - (Optional) Click in the **Family Namespace** field of the new row and type the family namespace of the family.

   - Choose the family **Type** (**Family**, **Summary Family**, or **Package Family**).

   - Enter data for all fields in a family by clicking in each column in turn, and then selecting or typing the required value. You can also edit data of an existing family.

     - If you create a Boolean family, a new row is automatically created and displayed below the family row. The new row represents a Boolean value.

     - If you create a unit of measure for the family, it is read-only and you cannot subsequently modify it.

   - Change the values in the **Sequence** column to reorder families.

You can also paste or drag existing families and their features from a configurator context or dictionary. However, if you try to paste or drag legacy data, you are not able to save any changes.

You can select all families in a group and all features in a family for allocation when you right-click the group or the family, respectively, and choose **Expand and Select**.

8. Click **Save** on the main toolbar to save all changes.

   Teamcenter displays the name and description in nonitalic text, indicating the family is saved.

If needed, you can release and revise a family. But you cannot revise a group. The system always displays groups because they are excluded from the revision-based search. You can add new features to a family even if that family is released.

If you copy a configurator context or dictionary by saving it under a new name (that is, by selecting the **Save As** command), Teamcenter does not carry forward the variant data.

Define features

You can create features for families in the context of the configurator context or dictionary. Features can be reused in multiple product structures and application models (product designs or partition templates).

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas. If needed, you can rename features, but their IDs remain unchanged.

1. Open the configurator context or dictionary in Product Configurator and ensure the **Variability Explorer** view is active.

   Teamcenter displays any existing features for the families in the tree table.

   You can add new features to a family even if that family is released.

2. Select the family for which you want to add a feature and click **Add Feature**.

   Teamcenter adds a new empty row as the child of the selected family and expands the family if necessary.

   When you add a feature to a dynamic family, the system only allows you to add a standalone feature.

3. Click in the **ID** field of the empty row and type the ID of the feature. The chosen ID must be unique for the current family; however, the same ID may be used in different families for the same configurator context or dictionary.

   For non-string families, you must assign an ID based on the data type of the family. The ID cannot be generated automatically. Only the ID of an authored feature is allowed in the configurator expression. For example, if the feature is authored with the ID of **4cm**, then all expressions that reference that value must contain **4cm**. An expression with IDs of **4** or **4 cm** is not supported.

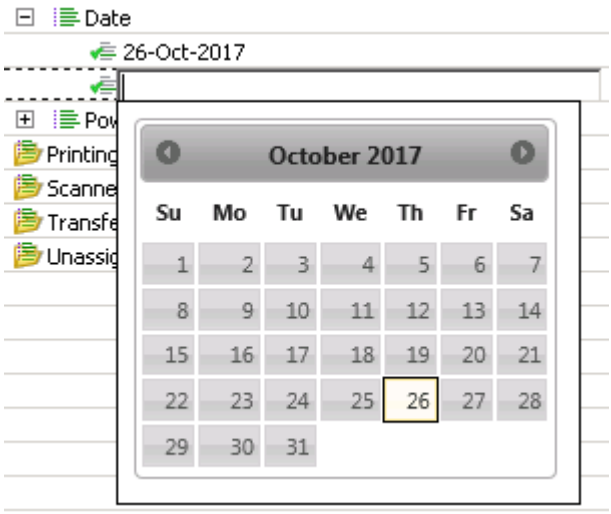
   Teamcenter adds the ID in italics, indicating the entry is not yet saved.

4. Click in the **Description** field of the empty row and type the description of the feature. Then, press Enter or click the next row.

   Teamcenter adds the description in italics, indicating the entry is not yet saved.

5. If a family is defined as date, perform the following steps to add the date value:

   - Click the newly added cell to display the calendar.



   - Set the required date.

     The calendar initially displays the current day, month, and year.

6. Click **Save** on the main toolbar to save all changes.

   Teamcenter displays the ID in nonitalic text, indicating the feature is saved.

ID	Family Namespace	Type
020420-Classic Car		Configurator Context
Body		Group
Body Type	Teamcenter	Family
Coupe		Feature
Hatchback		Feature
Sedan		Feature
Sport		Feature
Transmission		Group
Transmission Type	Teamcenter	Family
Wheels		Group
Unassigned Families		

7. As needed, click **Validate Variant Options** to validate a selected feature, on demand, to check for data consistency.

Alternatively, you can select a feature and choose **Validate Variant Options** from the shortcut menu.

If the validation fails for a feature, that feature is shown in red and the validation message appears. The failed feature displays the tooltip with the violation details. The tooltip lists object details that caused this violation, such as specific features or configurator rules.

Prior to save, you can change or edit an existing feature by clicking the field corresponding to the feature, then typing in the required new value.

You can copy features from an existing family associated with a configurator context or dictionary in one **Variability Explorer** view, and then paste them onto another configurator context or dictionary in another **Variability Explorer** view. You can also drag and drop features to achieve the same results. However, you cannot save legacy variant features copied in this way. Teamcenter checks if the feature's family is allocated to the target context. If yes, then it allocates the feature.

A standalone feature can be grouped under multiple dynamic families.

You can select all features in a family for allocation when you right-click the family and choose **Expand and Select** .

Revise configurator objects

Configurator features, configurator rules, or configurator allocation can be revised. In addition, you can revise a family but cannot revise a group.

Object	Can be revised?	Values that can be changed
Feature	Yes	None.
Group	No	Not applicable.
Family	Yes	**Name**, **Description**, **Optional**, **Minimum**, **Maximum**, **Sequence**, and custom values.
Product Model	Yes	**Name**, **Description**, and **Sequence**.

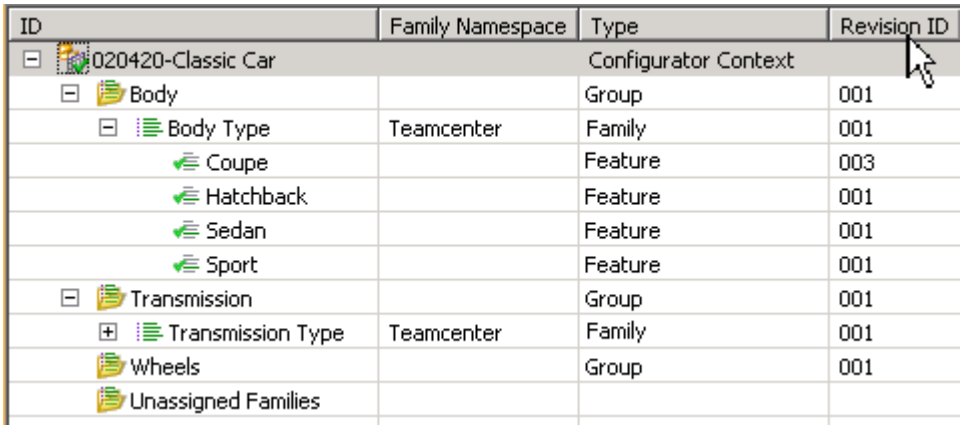
The following example shows how to revise a feature.

1. In the **Variability Explorer** view, select the feature you want to revise and choose **File**→**Revise**.

   Teamcenter displays the **Revise** dialog box.

2. (Optional) Clear the check box that checks out the new revision on creation and then click **Yes**.

   You can add the **Revision ID** column to view the feature revisions.



ID	Family Namespace	Type	Revision ID
020420-Classic Car		Configurator Context	
Body		Group	001
Body Type	Teamcenter	Family	001
Coupe		Feature	003
Hatchback		Feature	001
Sedan		Feature	001
Sport		Feature	001
Transmission		Group	001
Transmission Type	Teamcenter	Family	001
Wheels		Group	001
Unassigned Families			

Teamcenter creates a new revision of the feature and optionally checks it out. It pastes the new revision in the **Newstuff** folder. The effectivity of the new revision is the same as the previous revision.

Using the **Revise** command, you can also discontinue a revision of a configurator object, such as a family (standard or dynamic), model family, feature, configurator allocation, model, and rule.

Create a summary family in a configurator context or dictionary

You can create summary families for a configurator context or dictionary.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas. If needed, you can rename configurator objects, but their IDs remain unchanged.

1. Search for and open an existing configurator context or dictionary in Product Configurator and ensure the **Variability Explorer** view is active.

   Teamcenter displays any existing features for the families in the tree table.

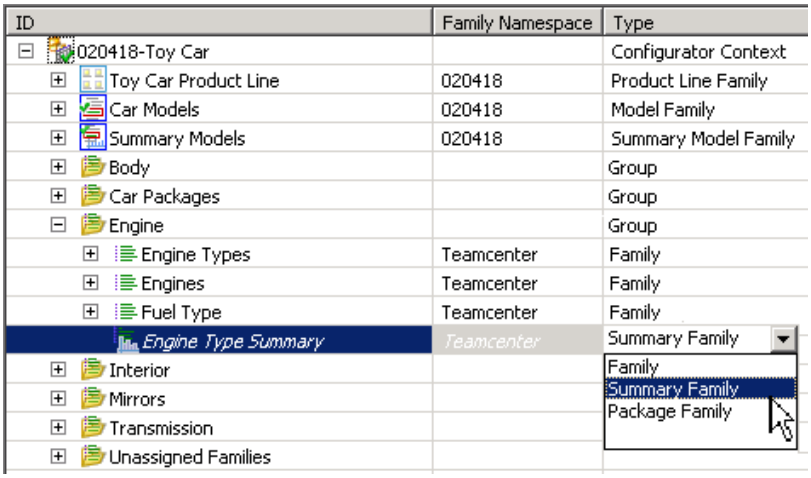
2. Select the group for which you want to add a summary family, and click **Add Family** .

   Teamcenter adds a new empty row as the child of the selected group and expands it if necessary.

3. Click in the **ID** field of the empty row, and type the ID of the summary family. The chosen ID must be unique for the current configurator context.

   Teamcenter adds the ID in italics, indicating the entry is not yet saved. An asterisk (*) next to the name of the view also indicates there is unsaved data.

4. Click **Type** for the new row, and select **Summary Family** from the list of available types.



5. (Optional) Click **Description** for the new row, and type the description.

   The **Feature Data Type**, **Minimum Value** and **Maximum Value** values are not modifiable.

6. (Optional) You can change the **Multi-select** fields or set the **Sequence** value for the newly created summary.

7. Click **Save** .

   Teamcenter displays the summary family icon and name and description in nonitalic text, indicating the group is saved.

   The summary family is created.

8. To remove a selected summary family, right-click and choose **Cut**.

   Deallocate the dependent objects prior to deallocating the summary family.

Define feature summaries

You can create feature summaries for summary families in the configurator context or dictionary.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas. If needed, you can rename configurator objects, but their IDs remain unchanged.

1. Search for and open an existing configurator context or dictionary in Product Configurator, and ensure the **Variability Explorer** view is active.

   Teamcenter displays any existing features for the families in the tree table.

2. Select the summary family for which you want to add a feature summary, and click **Add Feature** .

   Teamcenter adds a new empty row as the child of the selected family and expands the family if necessary.

3. Click in the **ID** field of the empty row, and type the ID of the feature summary. The chosen ID must be unique for the current configurator context; however, the same ID may be used in different families for the same configurator context or dictionary.

   Teamcenter adds the ID in italics, indicating the entry is not yet saved. An asterisk (*) next to the name of the view also indicates there is unsaved data.

4. Click **Save** .

   Teamcenter displays the feature summary icon and name and description in nonitalic text, indicating the value is saved.

   The feature summary is created.

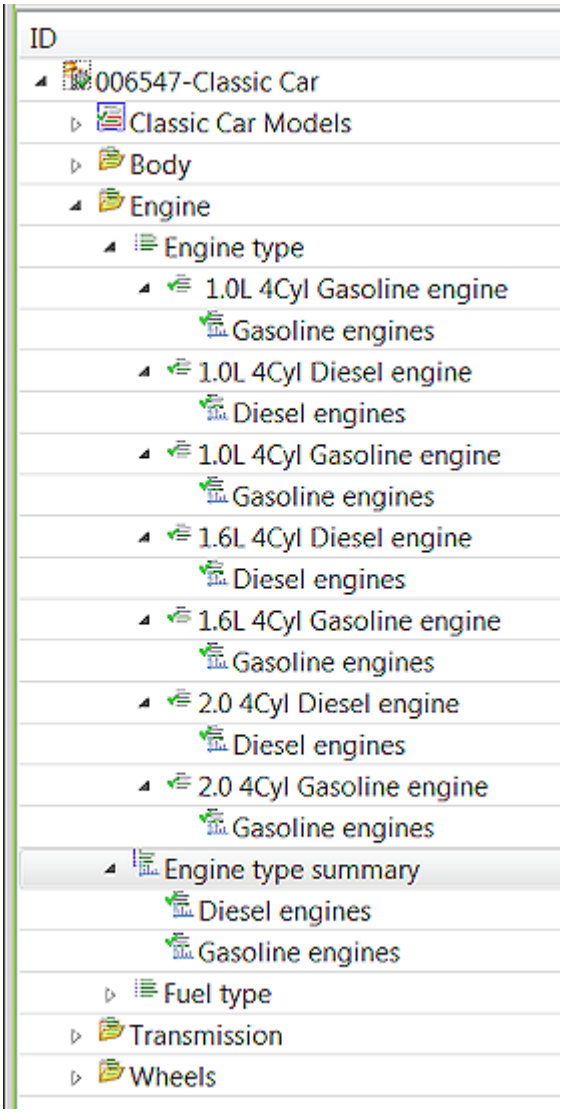
5. To remove a selected feature summary, right-click and choose **Cut**.

6. Define the new feature summary by doing one or more of the following:

   - Copy features and paste them onto the feature summary. You can copy and paste multiple features on the feature summary.
   - Drag an existing feature onto the feature summary.
   - Remove a selected feature by right-clicking and choosing **Cut**. If the feature summary is used in rules, an error message is displayed.

   You must have write access to the summary family to add or remove members. Features may belong to more than one feature summary. Features may also be from multiple multiselect or mutually exclusive families.

   For example, **1.0L 4Cyl Gasoline engine**, **1.6L 4Cyl Gasoline engine** and **2.0L 4Cyl Gasoline engine** are summarized by the **Gasoline Engines** summary family. Teamcenter displays **Gasoline Engines** as the child element of **1.0L 4Cyl Gasoline engine**, **1.6L 4Cyl Gasoline engine**, and **2.0L 4Cyl Gasoline engine** features. The **Diesel Engines** feature summary contains the **1.0L 4Cyl Diesel engine**, **1.6L 4Cyl Diesel engine**, and **2.0L 4Cyl Diesel engine** features.



You can copy a feature summary or a summary family on an existing family associated with a configurator context or dictionary in one **Variability Explorer** view and then paste them onto a family associated with another configurator context or dictionary in another **Variability Explorer** view. You can also drag and drop a feature summary or a summary family to achieve the same results. You must have write access to the members of that target dictionary or configurator context.

If features in the dictionary were initially summarized in the configurator context, you can only view those feature summaries that are already allocated to the dictionary and summarize the feature.

Create package families and feature packages for a configurator context or dictionary

You can create package families and assign feature packages for a configurator context or dictionary.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas. If needed, you can rename configurator objects, but their IDs remain unchanged.

1. Open the configurator context or dictionary in the **Variability Explorer** view.

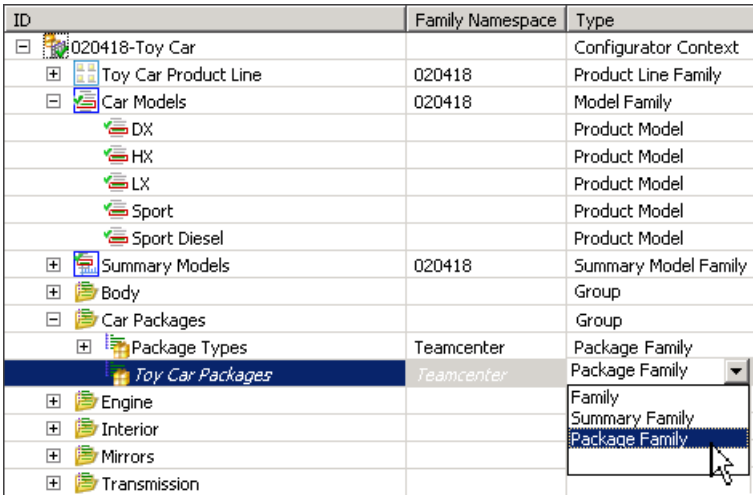
   Teamcenter displays any existing features for the families in the tree table.

2. Select a group. The selected group will contain the package family and feature packages.

3. Click **Add Family** to add a new family to the selected group.

   A new empty row appears as the last child of the selected group, and the group is expanded if necessary. When you type the ID of the family, the chosen ID must be unique for the current configurator context or dictionary.

4. Click **Type** for the new row, and select **Package Family** from the list of available types.



   If your administrator created your customer-specific types of packages, you can specify the custom type.

5. Select the package family for which you want to add a feature package, and click **Add Feature**.

   Teamcenter adds a new empty row as the child of the selected package and expands it if necessary.

6. Click in the **ID** field of the empty row, and type the ID of the feature package. The chosen ID must be unique for the current family; however, the same ID may be used in different families for the same configurator context or dictionary.

   Teamcenter adds the ID in italics, indicating the entry is not yet saved.

7. Click **Save**.

Teamcenter displays the package family icon and name and description in nonitalic text, indicating the family is saved.

The package family is created. The feature packages are also saved, and displayed with the icon .

8. To remove a selected feature package, right-click and choose **Cut**.

   Deallocate the dependent objects prior to deallocating the package family.

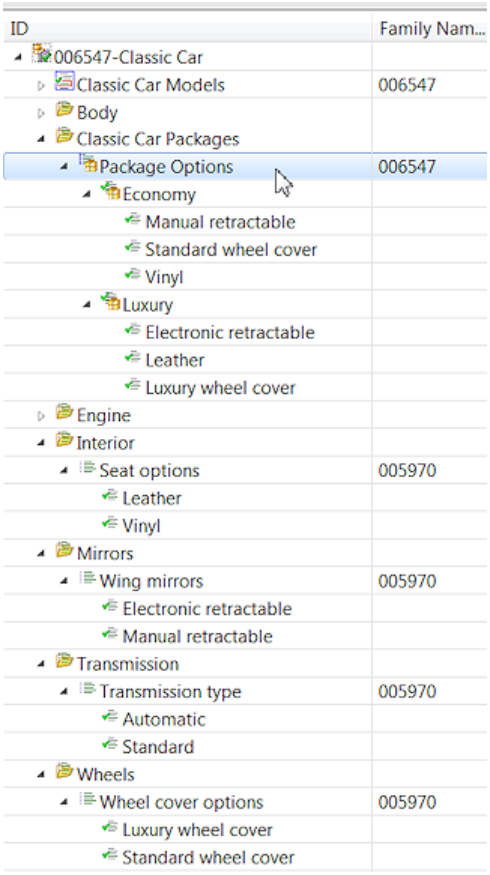
9. Define feature packages by doing one or more of the following:

   - Copy features and paste them onto the feature package.
   - Drag an existing feature onto the feature package.
   - Remove a selected feature by right-clicking and choosing **Cut**.

   Feature packages can only be created in the package family. Summary packages or another feature packages cannot be added to a feature package. You must have write access to the feature package to add and remove members.

   You can assign features to a feature package from multiple mutually exclusive and multiple multiselect families. Multiple features *cannot* be copied from the single select family. Multiple features *can* be copied from one mutually exclusive family.

   For example, the **Economy** feature package contains the **Manual retractable** mirrors, **Standard wheel cover**, and **Vinyl** seats. The **Luxury** feature package contains the **Electronic retractable** mirrors, **Luxury wheel cover**, and **Leather** seats.



You can copy feature packages or a package family on an existing family associated with a configurator context or dictionary in one **Variability Explorer** view, and then paste them onto a family associated with another configurator context or dictionary in another **Variability Explorer** view. You can also drag and drop a feature package or a package family to achieve the same results. However, you cannot save legacy variant features copied in this way.

If you added a new feature package to the source dictionary, when allocating to the target dictionary that newly added feature package is allocated to the package family in the target only if that package family is already allocated to it.

10. Click **Save** on the main toolbar to save all changes.

Define business intents

You can define intents on configurator objects, such as features, families, configurator rules, and groups.

The following example shows how to add intents to a feature.

1. Open the configurator context or dictionary in Product Configurator and ensure the **Variability Explorer** view is active.

Teamcenter displays any existing features for the families in the tree table.

2. Right-click an object for which you want to define an intent and choose **View Properties**.

3. In the **Properties** dialog box, click **Check-Out and Edit**. Then, click **Yes** to confirm the check out.

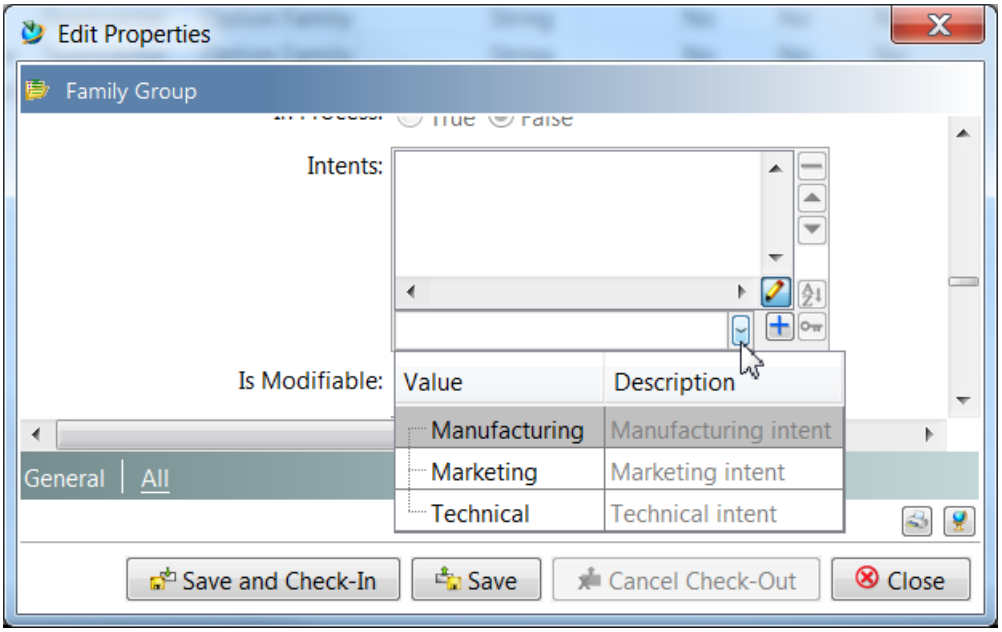
> Note:
>
> You must have write access to a selected object to be able to modify properties.

4. In the **Edit Properties** dialog box, click **All** to view all properties.

5. If the selected object does not have any assigned intents, click **Show empty properties**.

6. Click **Expand to modify**, and then ▾ to view a list of default intents.

> Note:
>
> You can also type the keywords in the text field to filter the list.

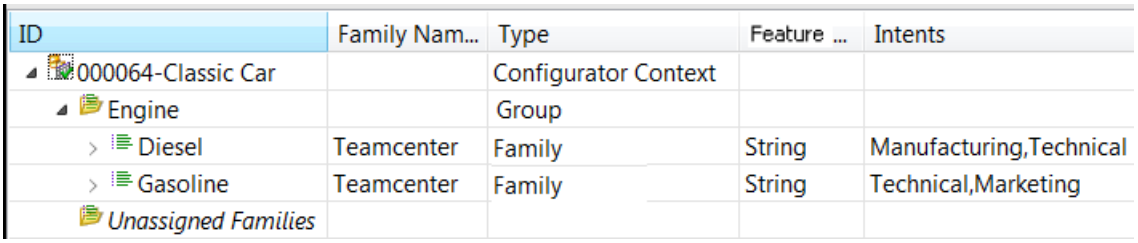


7. Choose an intent from the list and click **Add object** +.

   You can add more than one intent.

8. Click **Save and Check-In** to check in and save the changes.

9. The added intent values are shown in the **Intents** column in the configurator views, such as **Variability Explorer**.



ID	Family Nam...	Type	Feature ...	Intents
000064-Classic Car		Configurator Context		
Engine		Group		
Diesel	Teamcenter	Family	String	Manufacturing,Technical
Gasoline	Teamcenter	Family	String	Technical,Marketing
Unassigned Families				

> Note:
>
> You can manually add the **Intents** column.

6. Assign variability to product suite

Create a product line family, a model family, or a summary model family

To create a product model, product model family, summary model, or summary model family, you must have an appropriate license level. If the license is not available, the system generates an error during the creation the respective object.

1. Search for and open an existing configurator context in Product Configurator, and ensure the **Variability Explorer** view is active.

   Teamcenter displays any existing features for the families in the tree table.

2. Select the configurator context, and click **Add Model Family** .

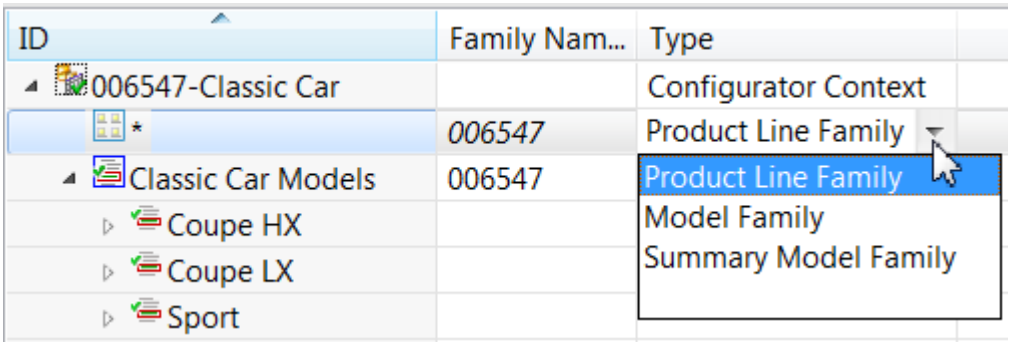
   Teamcenter adds a new empty row as the child of the selected family and expands the family if necessary.

3. Click in the **ID** field of the empty row, and type the ID of the model family. The chosen ID must be unique for the current configurator context.

   Teamcenter adds the ID in italics, indicating the entry is not yet saved. An asterisk (*) next to the name of the view also indicates there is unsaved data.

   If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas. If needed, you can rename configurator objects, but their IDs remain unchanged.

4. Click **Type** for the new row, and select **Product Line Family**, **Model Family**, or **Summary Model Family** from the list of available types.



If your administrator created your customer-specific types of model family options or summary model options, you can specify the custom type.

5. (Optional) Click **Description** for the new row, and type the description of the group.

   The **Feature Data Type**, **Optional**, **Free-form**, **Multi-select**, **Minimum Value**, and **Maximum Value** values are not modifiable.

6. (Optional) You can set the **Sequence** value for the newly created family.

7. Click **Save**.

   Teamcenter displays the model family icon, the product line family icon, or the summary model icon with a corresponding name and description in nonitalic text, indicating the group is saved.

ID	Family Namespace	Type
006547-Classic Car		Configurator Context
Summary Model Family	*006547*	*Summary Model Family*
Product Line Family	*006547*	*Product Line Family*
Classic Car Models	006547	Model Family

Create product models or summary models

You can create product models for product model families in the configurator context. Additionally, you can create summary models for summary model families.

1. Search for and open an existing configurator context in Product Configurator, and ensure the **Variability Explorer** view is active.

   Teamcenter displays any existing features for the families in the tree table.

2. Select the model family or a summary model family for which you want to add a product model or a summary model, respectively, and click **Add Model**.

   Teamcenter adds a new empty row as the child of the selected model family and expands the family if necessary.

   When you click **Add Model** within the context of a summary model family, Teamcenter recognizes that you intend to create a summary model.

> Note:
>
> To create a product model, product model family, summary model, or summary model family, you must have an appropriate license level. If the license is not available, the system generates an error during the creation of the respective object.

3. Click in the **ID** field of the empty row, and type the ID of the model. The chosen ID must be unique for the current configurator context; however, the same ID may be used in different configurator contexts.

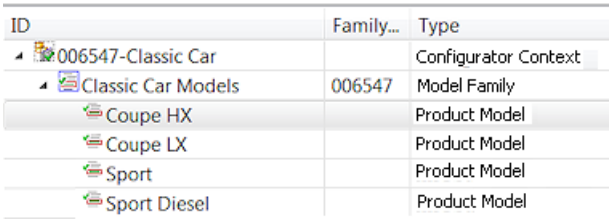
   Teamcenter adds the ID in italics, indicating the entry is not yet saved. An asterisk (*) next to the name of the view also indicates there is unsaved data.

4. Click **Save**.

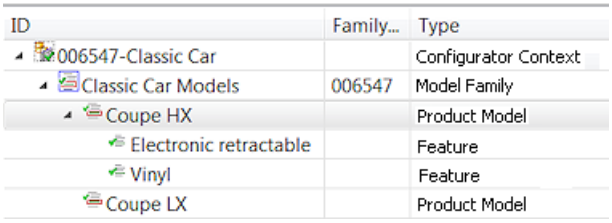
   Teamcenter displays the product model icon plus the name and description in nonitalic text, indicating the value is saved.

   The product model is created.

   For example, the **Classic Car Models** family contains the **Cobra HX**, **Cobra LX**, **Sports**, and **Sport Diesel** models.



   You can also copy features from existing families that are made available to that product model within the configurator context, and then paste them onto a product model to create a model package. You can also drag these features to achieve the same result.



   When a summary model family is created, Teamcenter displays the summary model icon plus the name and description in nonitalic text, indicating the value is saved.

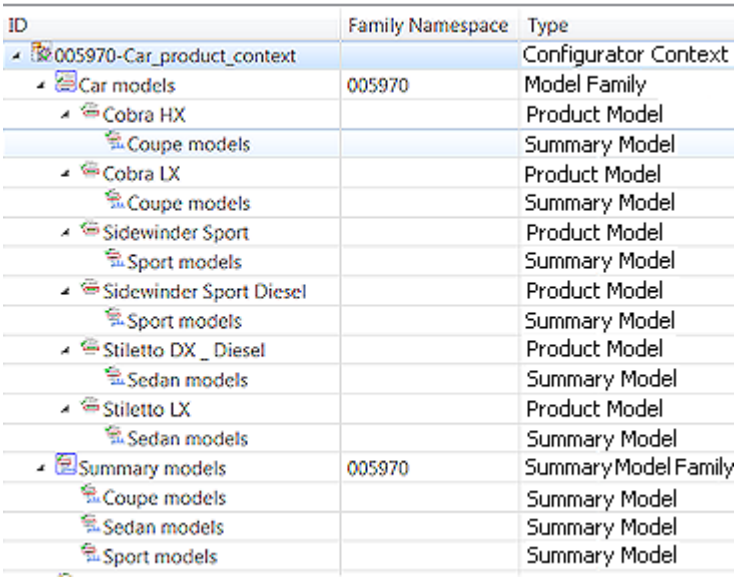
5. Summarize existing product models with a summary model by doing one or more of the following:

- Copy the models and paste them onto the summary model.
- Drag a product model onto the summary model.
- Remove a selected summary model by right-clicking and choosing **Cut**.

You can summarize multiple summary models with the same product model.

> Note:
>
> You cannot allocate features or families to the summary model.

For example, **Cobra HX** and **Cobra LX** are summarized by the **Coupe models** summary model. Teamcenter displays **Coupe models** as the child element of **Cobra HX** and **Cobra LX** models.

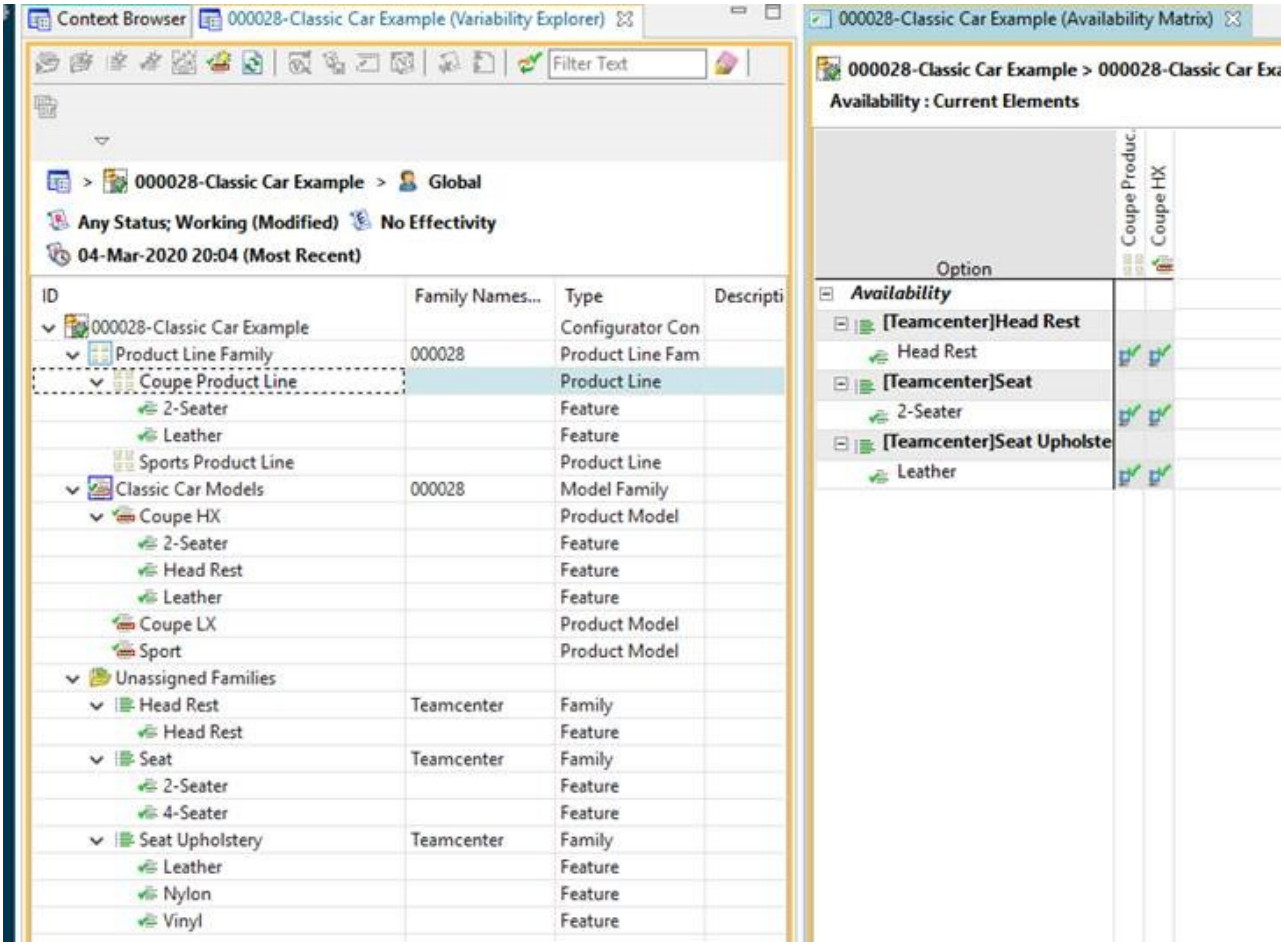


ID	Family Namespace	Type
005970-Car_product_context		Configurator Context
Car models	005970	Model Family
Cobra HX		Product Model
Coupe models		Summary Model
Cobra LX		Product Model
Coupe models		Summary Model
Sidewinder Sport		Product Model
Sport models		Summary Model
Sidewinder Sport Diesel		Product Model
Sport models		Summary Model
Stiletto DX _ Diesel		Product Model
Sedan models		Summary Model
Stiletto LX		Product Model
Sedan models		Summary Model
Summary models	005970	Summary Model Family
Coupe models		Summary Model
Sedan models		Summary Model
Sport models		Summary Model

6. If a product line exists, you can copy or drag features to that product line to add them as product line members.

   To create an implied hierarchy, when adding product line members, ensure to only add features that are members of product models or vice versa in that configurator context.

   To maintain the integrity of the implied hierarchy, you must implement processes to avoid undesirable effects, such as a broken or invalid hierarchy, as the system does not perform any validations during authoring.



> Note:
>
> In a given product line hierarchy, shared members should not be removed without understanding the implications because it may cause a broken or invalid hierarchy.

7. (Optional) You can set or update product line or model effectivity dates, units, or ranges.

   The system validates that the product line effective date range is narrower than or equal to the combined effectivity of its members. If this validation fails, you see an error message.

Add an image to a configuration object

You can add custom images to configuration objects, such as groups, features, product and summary models, and product lines). These images display next to the objects and make them easier to identify.

1. Create a jpeg dataset.

2. Copy the dataset.

3. Search for the object (group, feature, model, etc.) to which the image is to be added and edit its properties.

4. Right-click an object and choose **View Properties**.

5. In the **Properties** dialog box, click **Check-Out and Edit**. Then, click **Yes** to confirm the check out.

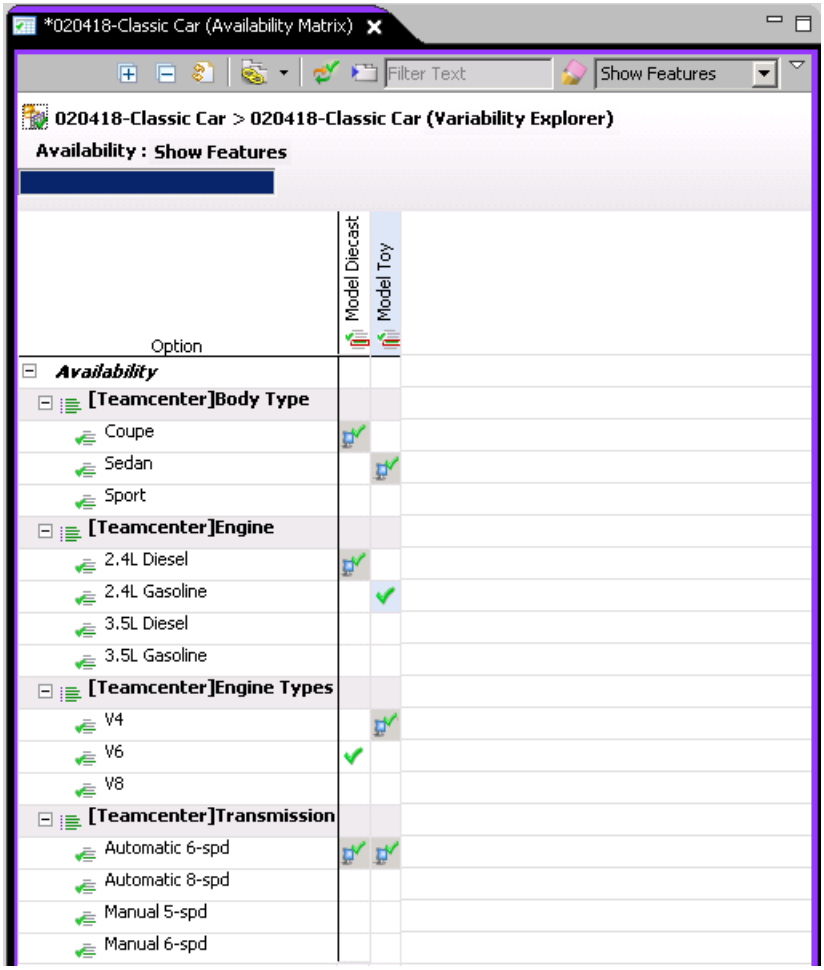
6. Scroll down and click **Show empty properties**.

7. Click to add or click + paste the copied dataset in the **Rendering** box.

8. Click **Save and Check-In** to check in and save the changes.

View and modify available features using the matrix

Use the **Availability Matrix** view to view and define features that are allowed for one or more product models, product lines, and summary models in a matrix style view. You can only choose from the allocated features in the configurator context.



If your system is configured with the global constant that points to **Name** as a primary business property, the matrix displays **Name** instead of **ID** in the corresponding rows.

1. Open the configurator context.

2. In the **Variability Explorer** view, select one or more product models, product lines, or summary models and click to open **Availability Matrix**.

   You can also right-click your one or more product models, product lines, or summary models and choose **Open with→Availability Matrix**.

   Revision rule from the **Variability Explorer** view controls the variance displayed in the availability matrix.

   Families, with an exception of the free-form family type, do not show the availability. If you select a product line or summary model, only its associated member models are displayed in the matrix.

   Feature summaries are not displayed in the matrix because you cannot define availability for summary. System does not perform availability checks on feature summaries, and hence, there is no need to create an availability rule for the feature summary.

   Teamcenter displays configured variability associated with the configurator context in a matrix view where models, product lines, summary models, and features are displayed in columns and rows, respectively.

   A system-assigned symbol is displayed next to the features that were made available.

   You can click additional product models, product lines, or summary models in the **Variability Explorer** view to add them to the matrix.

   The progress bar is shown below in a separate window. You can stop the data load at any time, if required.

3. By default, the variability is displayed using the **Current Elements** filter. The matrix displays only features available to at least one of the product models.

   Choose **Show Features** to view all features within the configurator context.

   By default, if availability does not exist, then Teamcenter opens availability with the **Show Features** filter.

4. You can choose the features to be available.

   - Click once to display a green check mark ✔ to make a feature available.

     Availability rules for new selections do not modify existing availability rules for the same feature.

When you choose features to be available for a product line, then the associated product models also display those features as available. The system creates the availability rule against the product line only.

You can also drag multiple features from the **Variability Explorer** view onto any model column in the **Availability matrix** view. Similarly, you can copy multiple features from the **Variability Explorer** view and paste them onto any model column in the **Availability matrix** view.

When availability is created using the matrix, no validation is performed against other constraints in the system. The system allows you to create availability rules for allocated features even if it conflicts with other constraints, for example, a feature may be excluded for a selected model. Thus, selecting a feature in the matrix is not a guarantee that the configurator context can be configured with this feature. Other constraint rules may exist that can cause validation failure even when availability rules exist.

Selecting a feature in the **Availability Matrix** view supersedes the system default for negative biased configurator context items by virtue of creating an availability rule for this feature and the selected product model. Therefore, availability rules are a necessary condition for a feature to be available when configuring a product represented by a negative biased configurator context item.

5. You can only remove availability ✔ for the available features in the product model prior to **Save**. After availability is saved, you can modify or delete a corresponding availability rule in the expression editor using the **Configurator Rules** view.

   After changing availability rules in the expression editor, you can view the results in the **Availability matrix** view.

6. Click **Save**.

   On the newly available features, a system-assigned symbol replaces a green check mark ✔ .

   A system-assigned symbol indicates that sufficient availability rules exist for a feature in a given product model based on variability information displayed in **Variability Explorer**. Inclusion and exclusion rules are not considered.

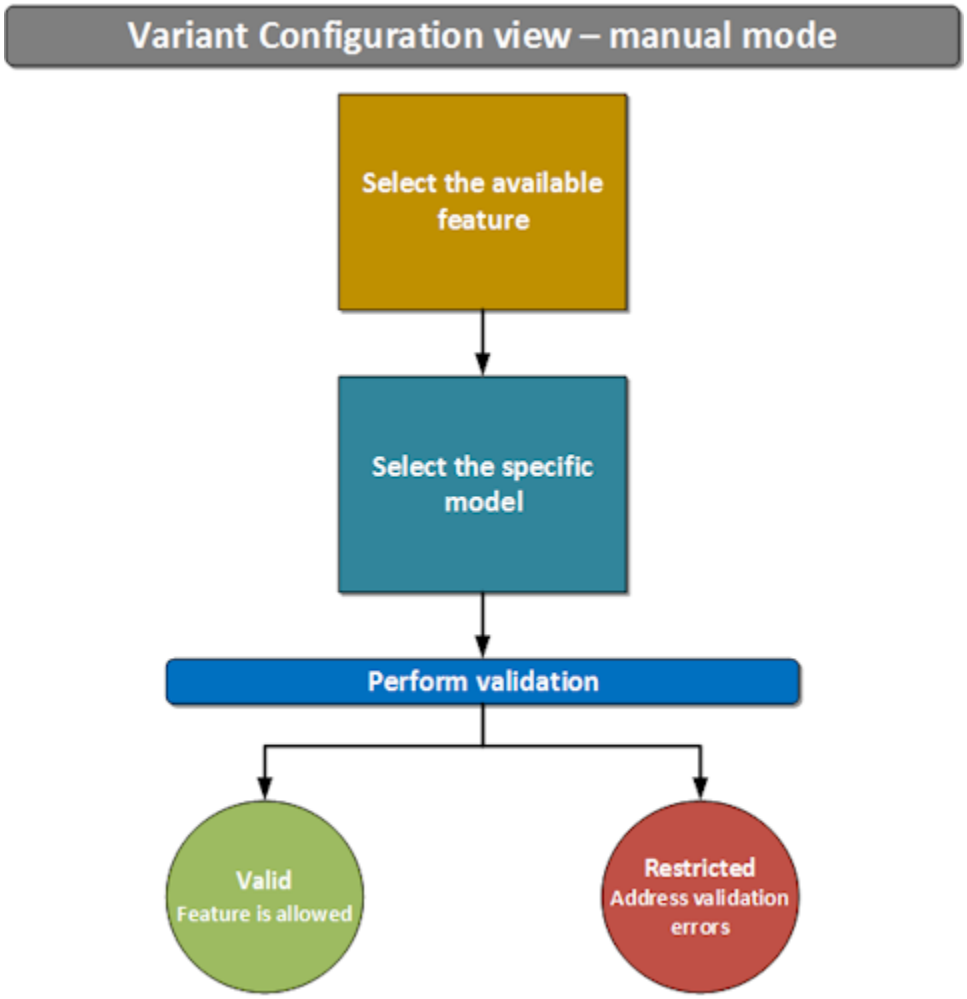
   Model members are implicitly available in a given model.

   Teamcenter generates availability rules for each feature as a **Subject** and **Condition** in the background. A feature is now available.

   To view availability rules, select your configurator context and view all availability rules in the **Configurator Rules** view.

7. A satisfiable availability rule is necessary but not sufficient for a feature to be allowed for a given model. If you plan to allow a specific feature for a given model, then a satisfiable availability rule must exist. You can create an availability rule for it in the matrix, but it may not be sufficient.

After creating the availability rule, it is recommended that you validate your selections using the manual validation in the **Variant Configuration** view. It allows you to check whether the feature is restricted by other constraints:



a. Select the feature you want to allow for a given set of models.

b. Select a specific model for which you want to allow the feature.

c. Validate this partial configuration.

- If it is valid, then the feature is allowed.
- If it is invalid, then the feature is restricted. Address the validation errors to find the constraints that restrict this feature for the given model.

d. Repeat the previous steps for every model where you want to allow this feature.

Tip:

To view newly created availability rules, you may need to clear the **Rule Date** in the breadcrumb. You work on a snapshot or a frozen set of the configurator data and use that data to get consistent results for a period of time. These results are not impacted by subsequent changes, such as additions or updates of features and rules.

7. Create configurator rules

Define configurator rules

You can author **configurator rules** for a set of features in a configurator context. Configurator rules (sometimes called *variant constraints*) define errors, warnings, and informative messages that Teamcenter generates for an invalid selection of features. Teamcenter supports the following types of Boolean variant constraints: inclusion, exclusion, availability, and default rules. Additionally, if your administrator has enabled the creation of dynamic families, you can also create exception rules for those dynamic families.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variability Explorer** view displays **Name** as a first column along with **ID** in the corresponding areas.

In the **Configurator Rules** view, the **Subject** condition, **Condition** condition, and **String Representation** display formula string as per the system configuration formula, such as display either with **Name** or **ID**.

In the **Variant Expression Editor**, the row header displays either **Name** or **ID**, as per the system configuration. But the column header is not impacted by this change.

Your administrator can add **ID** and custom properties, as needed, using the **Cfg0ExpressionGridColumnProperties** preference for the expression grid in the **Variant Expression Editor** view.

1. Open the configurator context or the dictionary in the **Variability Explorer** view, and then click **Open Configurator Rules view** to open the configurator rules editor.

   Teamcenter displays the selected editor.

   You can filter the list of displayed configurator rules to make it easier to locate the features you are interested in.

   You can click to select configurator features, models, or product lines to view existing configurator rules associated with them.

2. Limit the number and filter the list of displayed configurator rules by:

   a. Specifying values in the **Group ID**, **Family ID**, and **Feature ID** fields.

      An asterisk (*) can be supplied as a wild card to indicate any value.

If a family is defined as free-form, you cannot search for it using the asterisk * wild card in the **Feature ID**. To load the configurator rules that reference only free-form families, you must include the **Family ID** or **Group ID** in the search criteria.

b. Selecting one or more of the **Availability**, **Default**, **Exclusive**, or **Inclusive**, **Exception**, or **Free-form** check boxes.

To create availability rules, you must have an appropriate license level. If the license is not available, the system generates an error during the creation of this rule.

Exception rules are only available if your administrator enabled the creation of dynamic families and standalone features.

c. Selecting one or more of the **Info**, **Warning**, and **Error** icons.

3. Click **Search** to find existing rules based on your input search criteria.

The search result shows all rules, including the global rules, that match your search criteria. For global rules only, the **Is Global** column is set to **Yes**.

By default, the initial number of configurator rules displayed on a page is set to 100. Your administrator can change the default display by modifying the **Cfg0DefaultLoadCount** preference.

Rules are sorted by the configurator rule ID.

4. Move the vertical scroll bar or click at the bottom of the vertical scroll bar to load the next set of results.

By default, the subsequent number of loaded configurator rules displayed on the next page is set to 150. Your administrator can change the default by modifying the **Cfg0DefaultLoadCountIncreaseRate** preference.

5. Click **View menu**, and then select **Show Unreferenced Rules** to further filter the results to only show rules without any defined expressions, such as **Subject** or **Condition**.

To only view selected rules in the **Variant Expression** editor view, select one or more rules and click the **Show selected rules**.

6. Click in the **ID** field of the empty row and type the ID of the rule, or leave it blank so that the next available ID is automatically assigned.

7. Click the **Type** field in the same row.

Teamcenter displays a list of all configurator rule types.

Your administrator can create company-specific subtypes of configurator rules in Business Modeler IDE.

8. Select the **Type** (**Availability Rule**, **Default Rule**, **Exclusion Rule**, or **Inclusion Rule**, **Exception Rule**, or **Free-form**).

   You can select a custom configurator rule type, if your administrator created it in the system.

   You also can create availability rules for multiple models using a matrix style authoring in the **Availability Matrix** view.

9. Click the **Severity** field in the same row.

   Teamcenter displays a list of all severity types.

10. For inclusion and exclusion rules only:

    - Select a severity (**Error**, **Information**, or **Warning**).
    - Click the **Message** field of the same row and type the text of the message.

11. For availability rules, choose the **Error** severity level.

    Currently, the system evaluates all availability rules as if they have the **Error** severity level.

    Siemens Industry Software Inc. recommends to keep all availability rules at the **Error** severity level until the future release with an explicit support of multiple severity levels for availability rules.

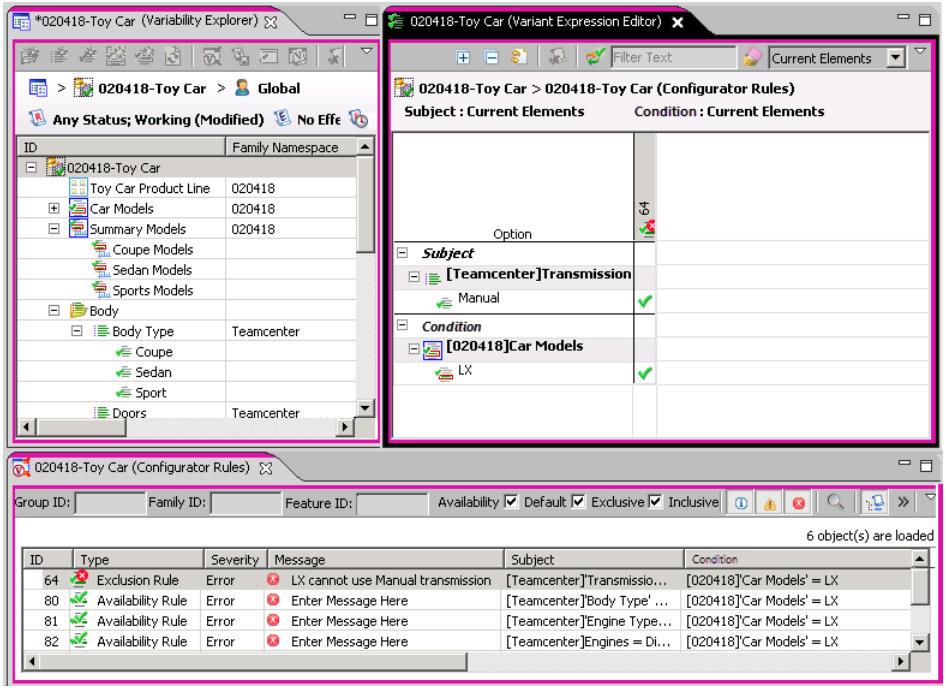
12. Press the **Enter** key or click anywhere in the view.

13. Click **Save**  on the main toolbar to save this newly created rule.

14. Click to display **Variant Expression Editor**.

15. Define a configurator rule in the grid view.

    Subject and **Condition** are listed in one grid view.



Choose **Current Elements**, **Show Families**, or **Show Features** to view only the selected features, selected features in families, or view all available features, respectively. Use **Filter Text** to selectively display only the features you are interested in.

You can view the full expression in **Configurator Rules View** by adding the **String Representation** column. Additionally, you can view the expression in the Teamcenter **Summary view**. You can also see a more detailed description of what that expression means in the **String Representation** box in the **Properties** window.

- When you define expressions in the grid view:
  - Selections are indicated by a check mark ✔ or crossed circle ⃠ in the cell.

    When you select a feature, that feature and the relevant column are highlighted in yellow and the expression text is updated to incorporate the change in the expression logic. The changes are highlighted visually until you save them to the system.
  - Selections in the same column are combined (AND function), except for selections in the same family, which are alternatives (OR function). You can add multiple columns for the same expression.
  - Multiple subexpressions are combined (OR function).
- You do not have to specify a model when constructing availability rules. You can have a model or other features in the availability statement. For example, you can make an engine available,

or you can make an engine available to transmission, or you can make an engine available to transmission for a specific model.

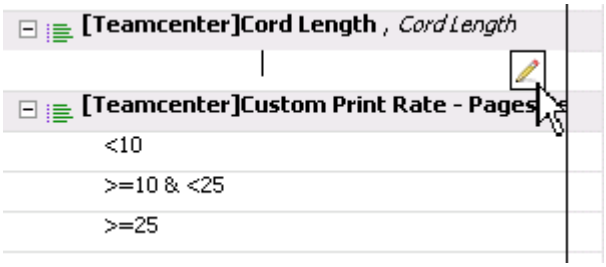
> Warning:
>
> Do not create configurator rules, variant conditions, or saved variant rules by combining a product model with a summary model or a product line.

- You can add a free-form text, numeric, or date value for a variant condition by performing the following steps:

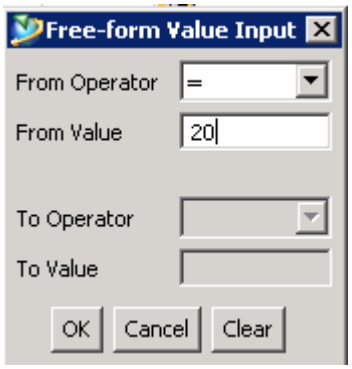
  a. Click the cell directly under the **Option** column containing the free-form variant.

     The edit pencil symbol appears.

  b. Click to enter the values.



  c. In the **Free-form Value Input** dialog box, type the values to set the variant condition.



     The **Free-form Value Input** dialog box displays **Name** instead of **ID** in the **From** and **To Value** dropdown lists, if the system is configured to point to **Name** for numeric and **Date** features. For a free-form family, the system displays user-assigned names.

  d. Click **OK**.

- You can add free-form string values for a variant condition by performing the following steps:

a. Click the cell directly under the **Option** column containing the free-form string family.

b. Enter values directly in the cell.

c. Click the cell to set the free-form variant condition for the feature.

d.

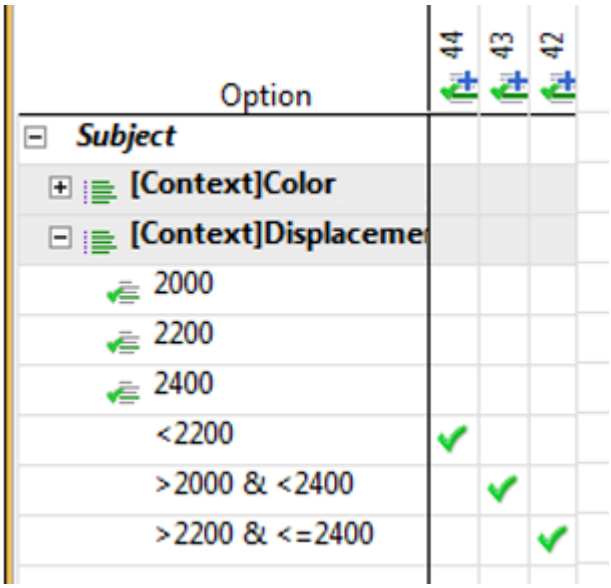
- You can specify operators with the free-form, numerical, and date type families when creating configurator rules.

  It allows you to efficiently author and maintain configurator rules when they apply to enumerated families. If you add more features to the family, the rules remain in effect because they contain expressions with operator values and ranges of features.

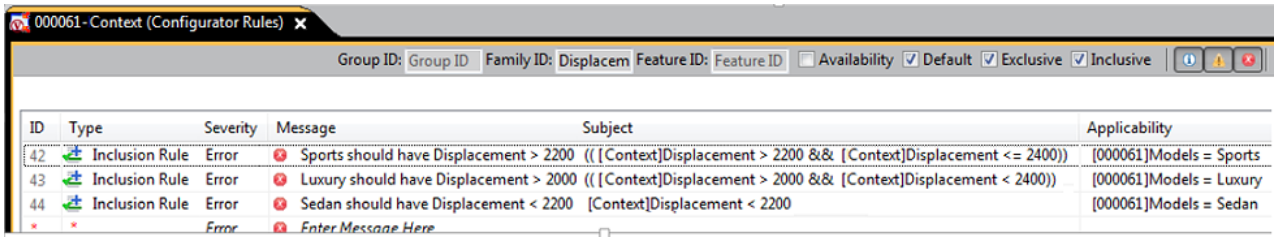
  To do so, use the pencil symbol in the blank row in the cell under the **Option** column for the corresponding family.

  You can only use = and != operators for text and free-form string features.

  For example, the **Displacement** family is an integer family with three enumerated features of **2000**, **2200**, and **2400**.

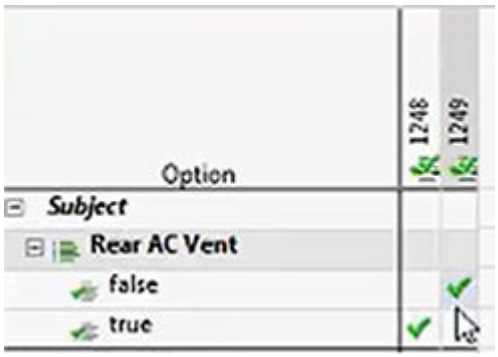


You can create expressions using range operators for features under the **Displacement** family. For example, the rule **44** states that **Displacement < 2200** for sedan models. Therefore, **2000** is the valid **Displacement** value.



> Note:
>
> If your system is configured with the global constant that points to **Name** as a primary business property, the search panel displays **Name** instead of **ID** in the corresponding areas.
>
> In the search panel in the **Configurator Rules** view, you can also search configurator rules by either feature, family, group **Name** or **ID**, but not both properties, based on the **Cfg0PrimaryBusinessRelevantAttribute** global constant setting.

- If you have a feature family with features of **true** and **false**, you can create and validate expressions referring to those features.



16. If required, split configurator rules.

17. Repeat steps 3 to 13 to create additional configurator rules, as needed.

18. Click **Save** on the main toolbar to save changes.

    If you hover over the rule, a tooltip with **Subject** and **Condition** of the selected expression appears.

    If the system is configured to point to **Name**, and then the tooltip displays **Name** instead of **ID** in the **Subject** and **Condition** expressions.

19. If Teamcenter detects any violations of configurator rules, the violation error message appears.

    The failed configurator rule is displayed with a warning symbol. If you hover over the failed rule, a tooltip with a violation message appears listing **Subject** and **Condition** of the selected expression. It also displays object details that caused this violation, such as specific features or configurator rules.

20. In **Variant Expression Editor** or the **Configurator Rules** view, select one or multiple rules and click **Validate Configurator Rules** to validate rules on demand.

> Tip:
>
> You may need to reset the **Rule Date** to today, clear the date or use the required date in order to view your availability rules. You work on a snapshot or a frozen set of the configurator data and use that data to get consistent results for a period of time. These results are not impacted by subsequent changes, such as additions or updates of features and rules.

Define free-form configurator rules

In addition to **configurator rules**, you can create complex free-form SMT-based rules using the SMT Lib scripting language. Using free-form rules, you can express restrictions and constraints that are not limited to a Teamcenter expression format for a standard configurator rule.

The ability to create free-form rules is disabled by default. Your administrator must use the **Cfg0EnableFreeFormRuleSupport** preference to enable this functionality. By default, it is set to **True**.

1. Open the configurator context or the dictionary in the **Variability Explorer** view, and then click **Open Configurator Rules view** to open the configurator rules editor.

   Teamcenter displays the selected editor.

2. Click in the **ID** field of the empty row and type the ID of the rule, or leave it blank so that the next available ID is automatically assigned.

3. Click the **Type** field in the same row.

   Teamcenter displays a list of all configurator rule types.

   Your administrator can create company-specific subtypes of configurator rules in Business Modeler IDE.

4. Select the **Free-form Rule** type.

5. Click the **Severity** field in the same row.

   Teamcenter displays a list of all severity types.

6. Press the **Enter** key or click anywhere in the view.

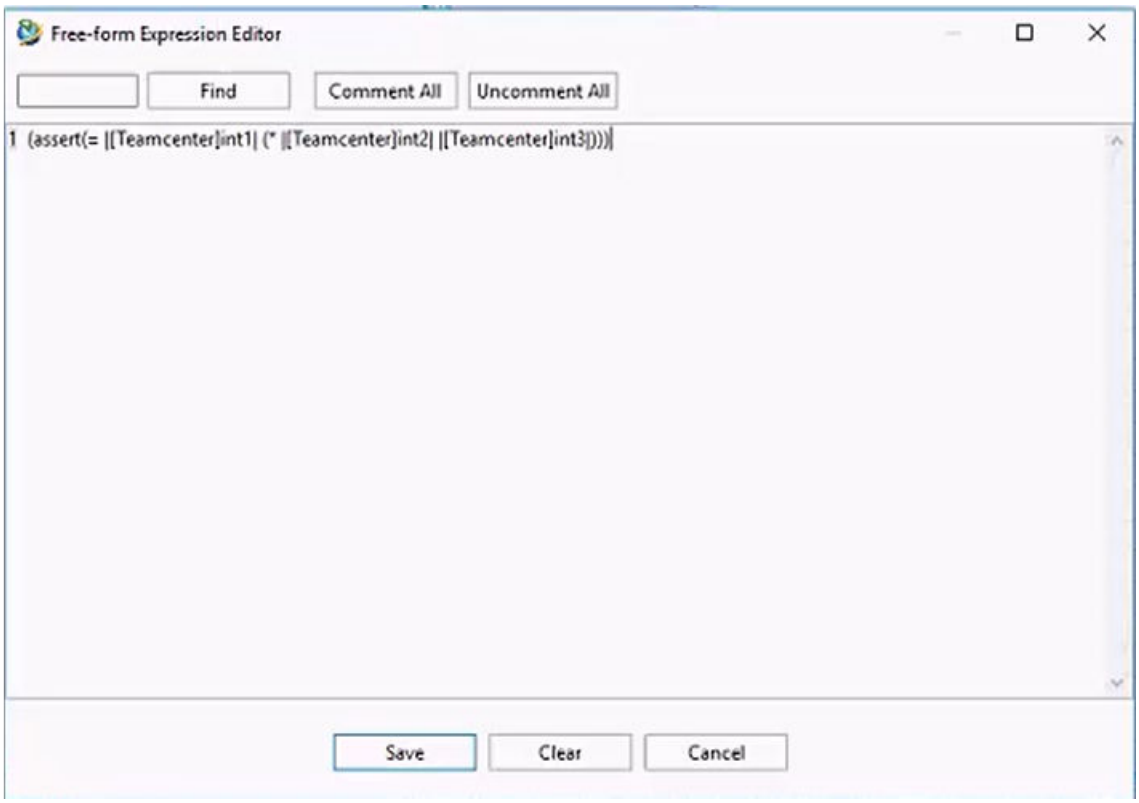
7. Click **Save** on the main toolbar to save this newly created rule.

8. You can either click **Open Free-form Expression Editor** or click the corresponding cell under the **Free-Form Expression** column to open the **Free-form Expression Editor** dialog box that allows you to create or modify free-form rules.

   For free-form rules, the command to display **Variant Expression Editor** is disabled.

   By default, the **Free-Form Expression** column is not visible. You can add columns by right-clicking the menu bar, choosing **Column**, and then modifying the list of displayed columns in the **Column Management** dialog box.

9. Define the complex rule expression using the industry standard language for a constraint solver. If the expression is incorrect, you cannot save and close the editor.



   The following is an example of a rule expression. The example in the screenshot and the example below defines the exact same constraint.

```
1       (assert
2           (let
3               (
4                   (TheaterCapacity |[Teamcenter]int1|)
5                   (RowCount |[Teamcenter]int2|)
6                   (SeatCount |[Teamcenter]int3|)
7               )
8               (= TheaterCapacity ( RowCount * SeatCount )
9           )
10      )
```

Click **Comment All** to comment out selected lines. The system ignores comments during processing. The comments are displayed in green. To include them, select one or many lines and click **Uncomment All**.

The system also displays the equivalent expression in the SMT format for standard configurator rules. However, you can only view but not modify the expression. The default and availability rules do not display the SMT expression.

10. Click **Save** on the main toolbar to save changes.

11. For the standard inclusion and exclusion rules only, you can view the SMT expression in the **Free-Form Expression** column. By default, the system converts inclusion and exclusion rules into a read-only free-form expression.

    You can copy a free-form expression generated from the include and exclude rule and use it as a base for the new free-form rule. Once that expression is used as a free-form rule, you can modify it as needed. If you make changes in the include or exclude rule free-form expression, the system does not update the **Subject** and **Condition** expressions accordingly. However, if you change **Subject** and **Condition**, the system updates the expression.

    To convert the include and exclude rule into a free-form rule.

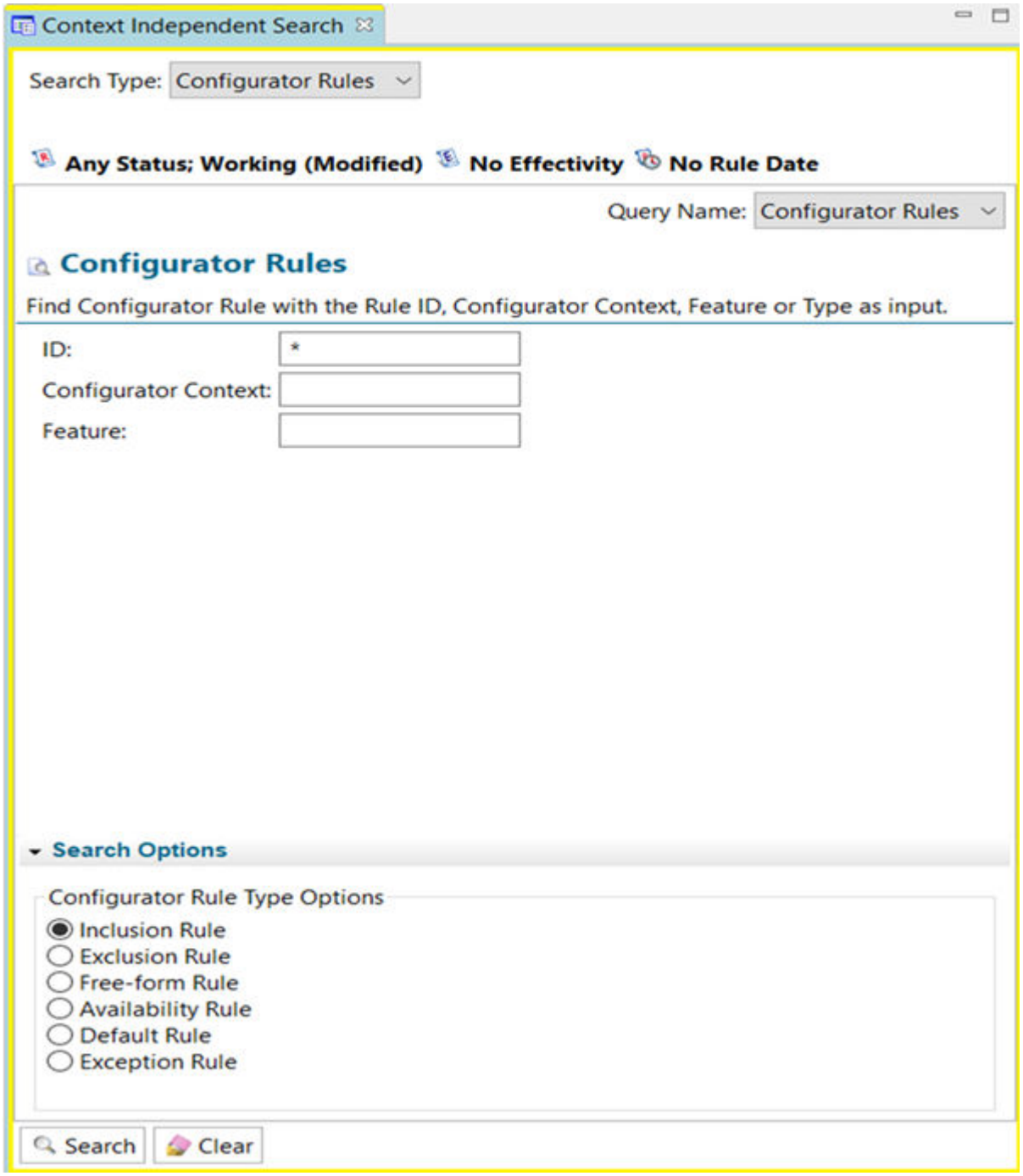
    - Locate the standard include or exclude rule from which you want to create a more complex free-form rule.
    - Click to open the **Free-form Expression Editor**.
    - Select and copy the expression generated by the system.
    - Create a brand new free-form rule and paste that expression into the **Free-form Expression** field.
    - Modify the rule as required.

- Delete the original standard include or exclude rule, if needed.

> Tip:
>
> You may need to reset the **Rule Date** to today, clear the date or use the required date in order to view your availability rules. You work on a snapshot or a frozen set of the configurator data and use that data to get consistent results for a period of time. These results are not impacted by subsequent changes, such as additions or updates of features and rules.

Search for configurator rules across configurator contexts

1. Search for existing configurator rules by clicking **Open Context Independent Search View** on the main toolbar.



You can search for only one type of rule at a time. To perform the search, specify the following:

- Rule **ID**
- **Configurator Context**
- **Feature**

- **Configurator Rule Type Options**, such as **Inclusion Rule**, **Exclusion Rule**, **Availability Rule**, **Default Rule**, **Free-form Rule**, or **Exception Rule**.

  You can add your company-specific configurator rule subtypes to the **Configurator Rule Type Options** pane by modifying the **PCA_RuleSearch_TypeOptions** preference.

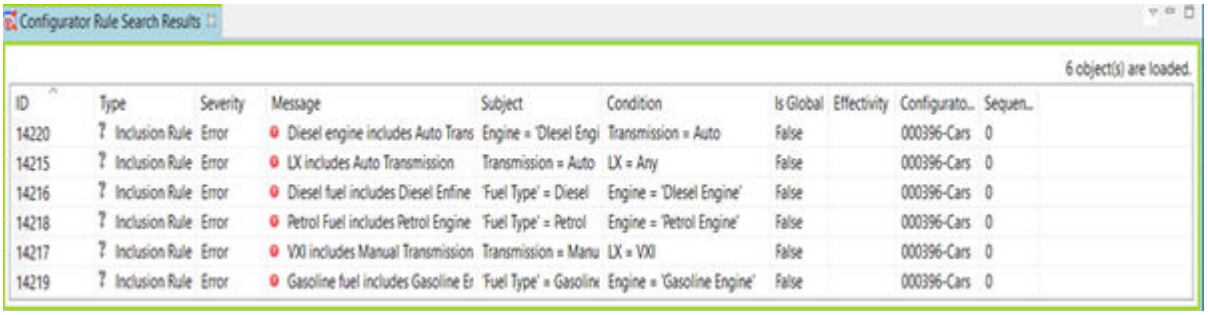
Additionally, you can choose the desired revision rule or the effective date to further filter your results.

In addition to a **Configurator Rules** query, your administrator can create additional search query types by creating a new saved query and adding it to the **PCA_RuleSearch_SavedQueries** preference.

You administrator can also add additional search criteria by modifying the **Configurator Rules** saved query and adding custom or removing existing properties.

Because some of the resulting rules may be related to different contexts, the **Configurator Rule Search Results** view contains a **Configurator Context** column that displays the source configurator contexts.

Teamcenter displays the results in the **Configurator Rule Search Results** view.



Copy configurator rules

1. Open your configurator context in the **Variability Explorer** view, and then click **Open Configurator Rules View** to open the configurator rules editor.

   Teamcenter displays the **Configurator Rules** view.

2. You can filter the list of displayed configurator rules to make it easier to locate the values you are interested in.

   In the **Configurator Rules** view, you can limit the number and filter the list of displayed configurator rules by:

   - Specifying values in the **Group ID**, **Family ID** and **Feature ID** fields.

- Selecting one or more of the **Availability**, **Default**, **Exclusive**, **Inclusive**, or **Free-form** check boxes.
- Selecting one or more of the **Info**, **Warning**, and **Error** icons.
- Click **Search** to refresh the list.

You can also locate configurator rules by using **Context Independent Search**.

> Tip:
>
> You can click to select configurator features, models, or product lines to view existing configurator rules associated with them.

3. Select the rule and choose **File** → **Save As**.

   You can also send your rules to My Teamcenter by right-clicking the rule and choosing **Send To** → **My Teamcenter**. Then, you can use the **Save As** command to duplicate configurator rules in My Teamcenter.

   Teamcenter displays the **Save As** information dialog box.

4. Assign a new ID and add description.

5. Click **Finish**.

   Teamcenter creates a copy of the selected configurator rule. The copied configurator rule is associated with the source configurator context.

   By default, the **Save As** behavior is that metadata is carried forward to the new object by default. Your administrator can define deep copy rules in Business Modeler IDE that designate any types of relationships that are carried forward from the existing object to the new object.

Share configurator rules across configurator contexts

1. Search for existing configurator rules by clicking **Open Context Independent Search View** on the main toolbar.

   Teamcenter displays the results in **Configurator Rule Search Results**.

2. In the **Configurator Rule Search Results** view, select one or more configurator rules that you are planning to associate with the target configurator context.

3. Open a separate window with the target configurator context.

4. From the **Configurator Rule Search Results** view, drag the selected rules onto the target configurator context.

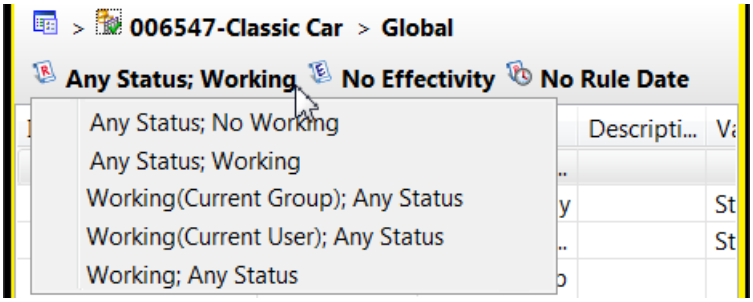
   You can also copy the rules from the **Configurator Rule Search Results** view and paste them onto the target configurator context.

   If any of the features used in the rule are not allocated or if the rule already exists in the target configurator context, Teamcenter displays a warning.

   You must have write access to the target configurator context to successfully perform this command.

Set a revision rule on a configurator context or dictionary

1. In the **Variability Explorer** view, select the configurator context or dictionary, you want to configure with a revision rule as the top line of the navigation tree. The current revision rule is shown at the top of the pane, for example, **Any Status, Working**.



> Note:
>
> If you have not set another revision rule, the default rule is applied.

2. To set a different rule, click the hyperlink and select a new revision rule from the dropdown list of available revision rules. The list contains only rules that can be applied to configurator contexts or dictionaries; other revision rules in the system are hidden.

   Teamcenter applies the selected revision rule, and refreshes the tree.

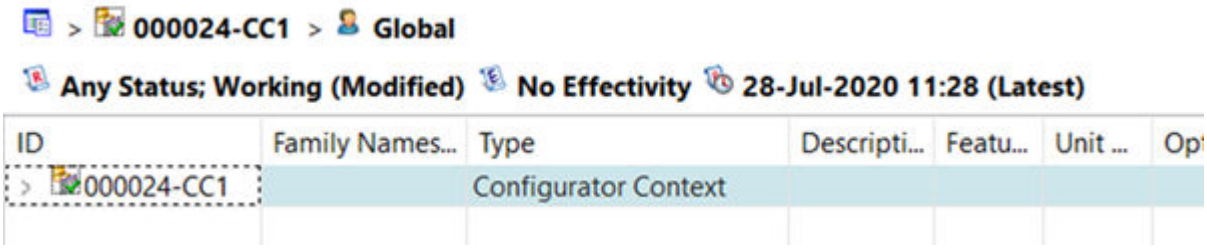
> Note:
>
> Your company-specific revision rules are created by an administrator.

Set or change a rule date for a configurator context or a dictionary in rich client

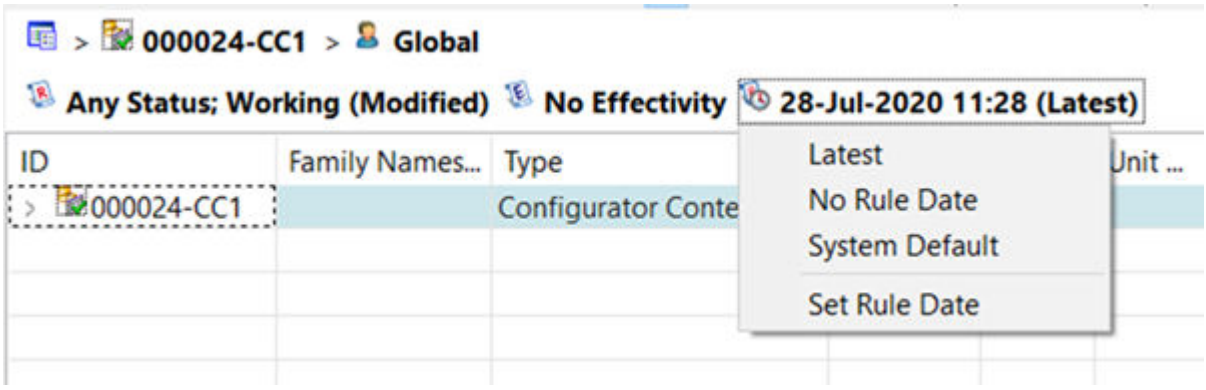
1. Open a dictionary or a configurator context in the **Variability Explorer**.

Teamcenter applies the current timestamp rule date with **Latest** to indicate that the system displays the latest available snapshot.

> Note:
>
> You can change the default rule date by using the **Cfg0DefaultRuleDateForPCA** preference.



2. Click the date if the rule date was previously set and choose **Latest**, **No Rule Date**, **System Default**, or **Set Rule Date**.



- Select one of the following:
  - **Latest** to load the latest available snapshot from the current system date and time.
  - **No Rule Date** to load the latest available snapshot in the system.

> Example:
>
> Consider a scenario where the configurator context is being edited concurrently by multiple users. When the **No Rule Date** option is chosen, the system considers all

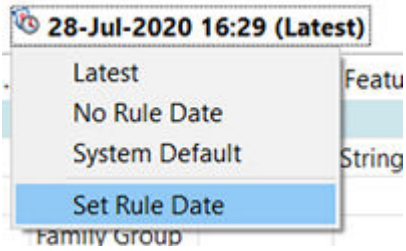
changes done till you select the **Validate**, **Expand**, or **Apply Configuration** command. Every solve action involves multiple database queries to get all the latest data. This option is not recommended in such a scenario as it results in performance degradation.

When you select the **Latest** option, the system considers all changes done till you open the **Variant Configuration** view. Therefore, only the first solve action involves multiple database queries to make sure that all the latest data has been used just before the first request. Thereafter, every solve request using the **Validate**, **Expand**, or **Apply Configuration** command, uses the data from the first solve action. The subsequent solve requests from the same **Variant Configuration** view does not lead to multiple database queries, which results in better performance. If any user creates or modifies the data after the first solve request, it is not considered for subsequent solve requests.

- **System Default** to load the snapshot as defined by your system administrator.

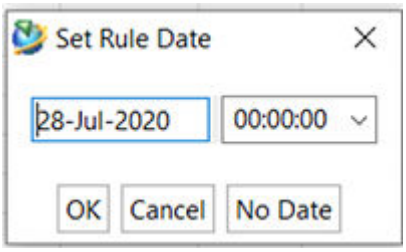
  For example, in this snapshot, Teamcenter applied the rule date as midnight of July 28, 2020 based on the offset calculation.

- **Set Rule Date** to set a specific date and time using date control.

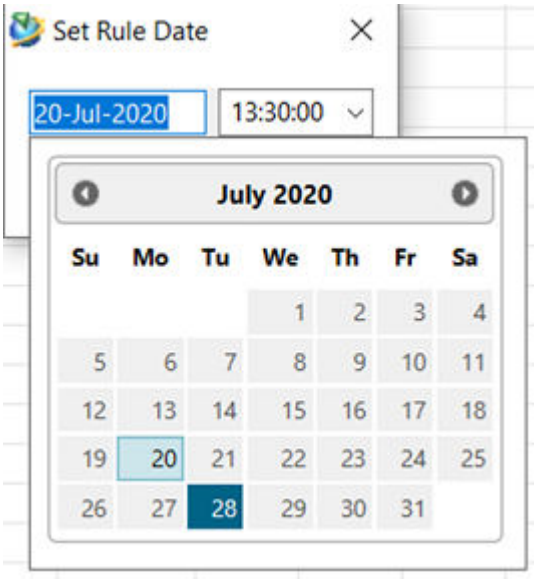


3. Set a specific date and time.

   a. To set the current date, click **OK** in the **Set Rule Date** dialog box.



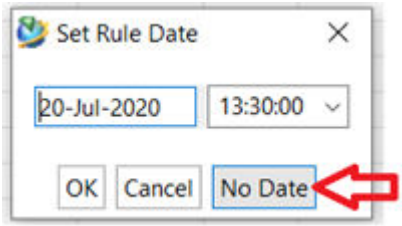
   b. To set a specific date, select a date from the calendar and select or type the time.



You can select time in seconds, but the breadcrumb date link does not display the seconds.

c. Click **OK** to set the specified date.

4. To clear the rule date and set the **No Rule Date** configuration, click **No Date**. You can switch to **No Rule Date** from the **Set Rule date** dialog box.



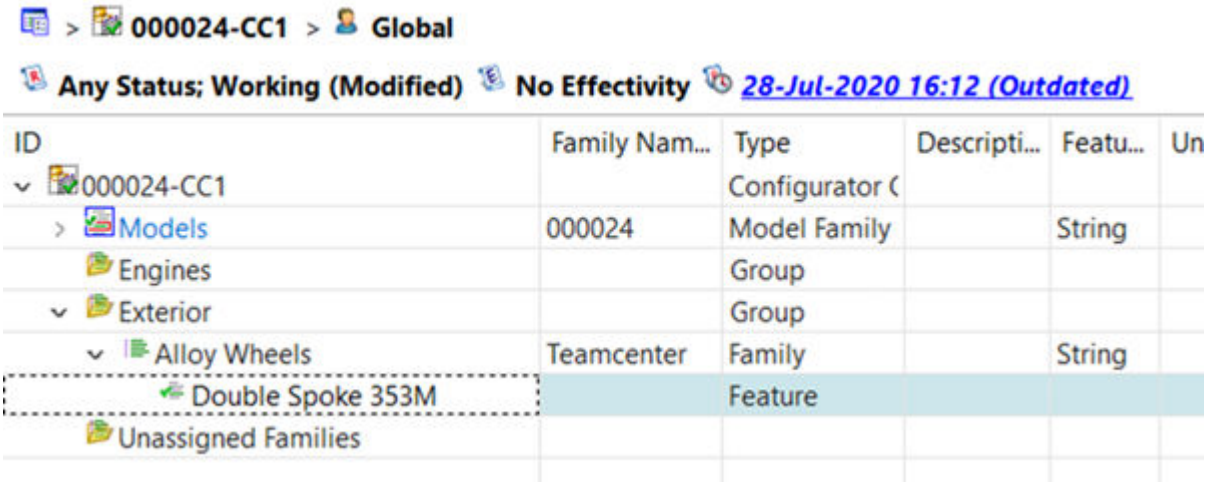
The rule date is set and the configurator context or dictionary is configured using the rule date.

When you assign an effectivity, the revision rule and rule date is also displayed and used as configuration criteria.

5. If a new family, feature, variability data, configurator rules, variant rules, or expression is authored for rules by using **Expression Editor** or availability is created by using the **Availability Matrix** view, the **Rule Date** link is displayed with **Outdated** appended to it.

   The string **Outdated** is an indication that any further action is performed using the old configurator snapshot. You cannot use these newly created objects for further workflows such as authoring variant expression or performing variant configurations.

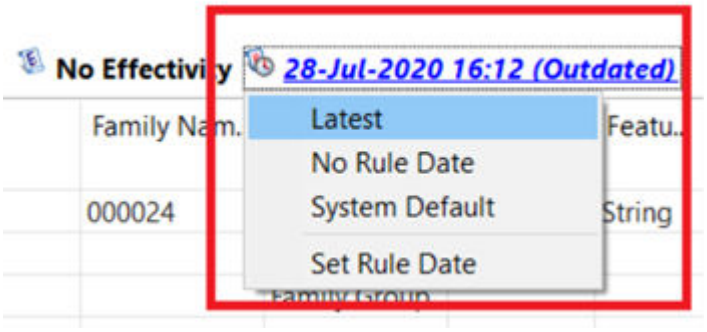
   In this example, the **Double Spoke 543M** alloy wheel was newly added. After you perform a save, the rule date is shown as **Outdated**.



6. To move from the **Outdated** state to any other rule date:

   a. Hover over the date. The tooltip informs you that the current view does not display all the newly created features and which action you can take.

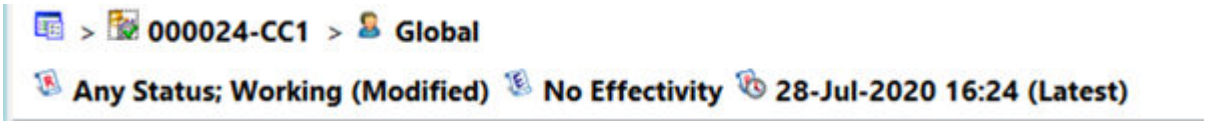
   b. You can switch to a rule date option by selecting one of the following options:



7. Click **Reload View** in the view toolbar to update the snapshot.

   Teamcenter applies the current timestamp as the rule date and reloads the view contents.

In this example, you reloaded the view at 11:42 am on April 12, 2017.

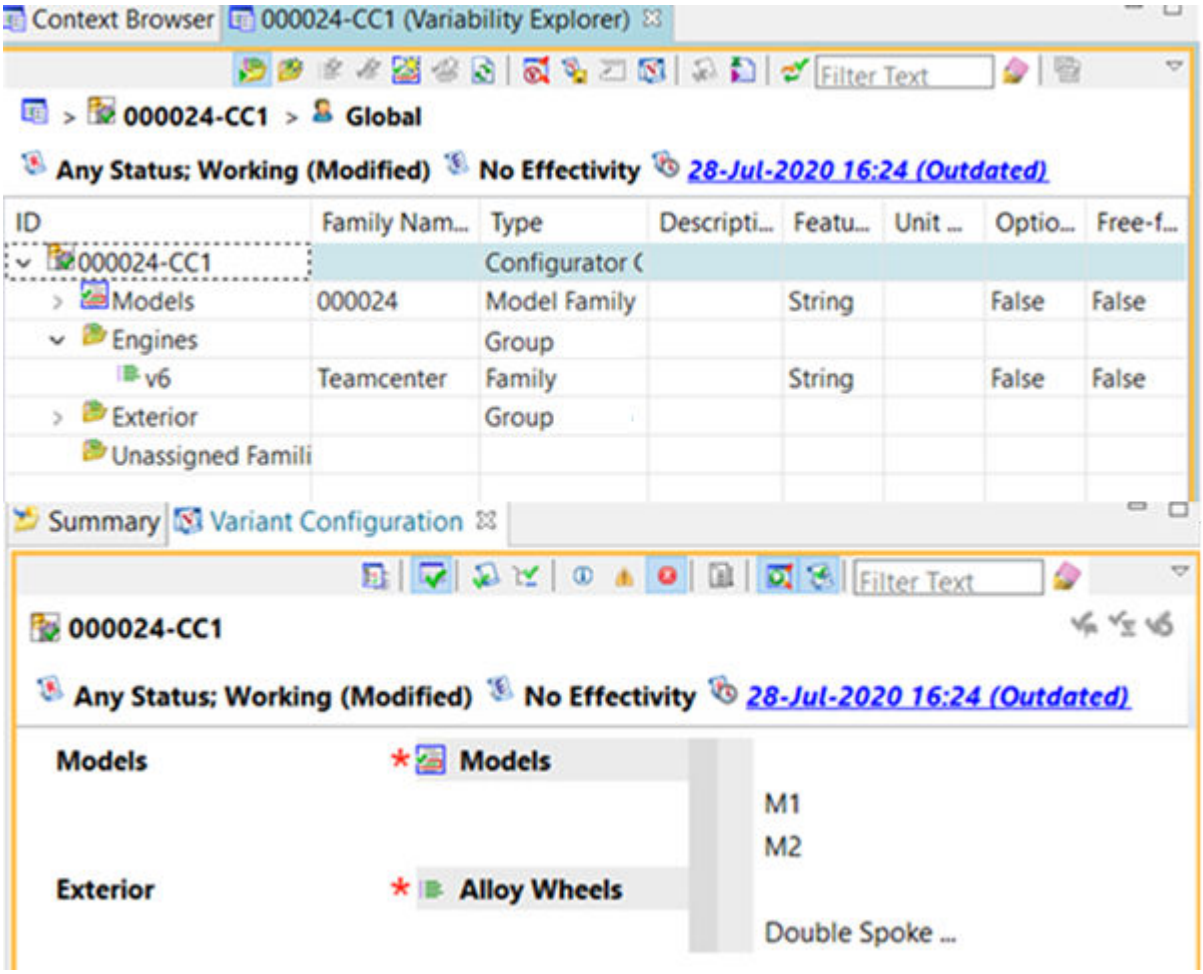


Teamcenter removes the style applied to the rule date in the previous step. The breadcrumb also contains **Latest** to indicate that it is the latest snapshot.

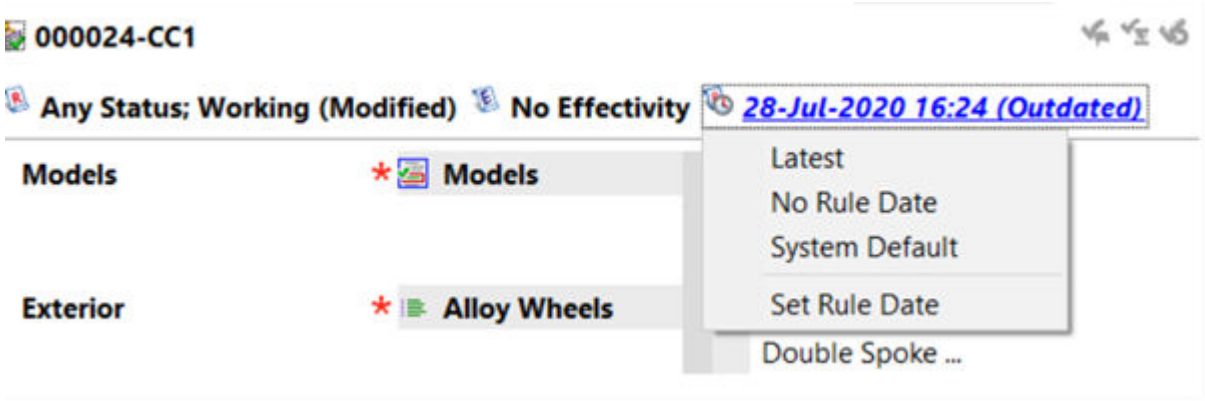
8. Set or change the rule date in the **Variant Configuration** view.

When the rule date is shown as **Outdated** in the **Explorer** view, you can open the **Variant Configuration** view without refreshing or updating the rules date in **Explorer** view. The rule date link continues to be displayed as **Outdated**.

In this example, a feature V6 engine is added in the **Explorer** view. Due to which, the rule date link in is displayed as **Outdated**. When you open the **Variant Configuration** view, the same rule date is displayed as **Outdated**. The **Outdated** link in the configuration view helps you in understanding why the feature V6 engine is not displayed in the **Variant Configuration** view.



You can change the rule using the rule date in the **Variant Configuration** view.



Configure features, rules, models, and product lines using effectivity through a configurator context or a dictionary

When you apply effectivity to a configurator context or dictionary, the features are configured according to the unit range or the date range you specify.

1. If needed, add the **Effectivity** column to view the effectivity.

   The columns displayed are configured by your administrator.

2. Right-click the configurator objects, such as features, rules, models, and product lines and choose **Open With→Effectivity View**.

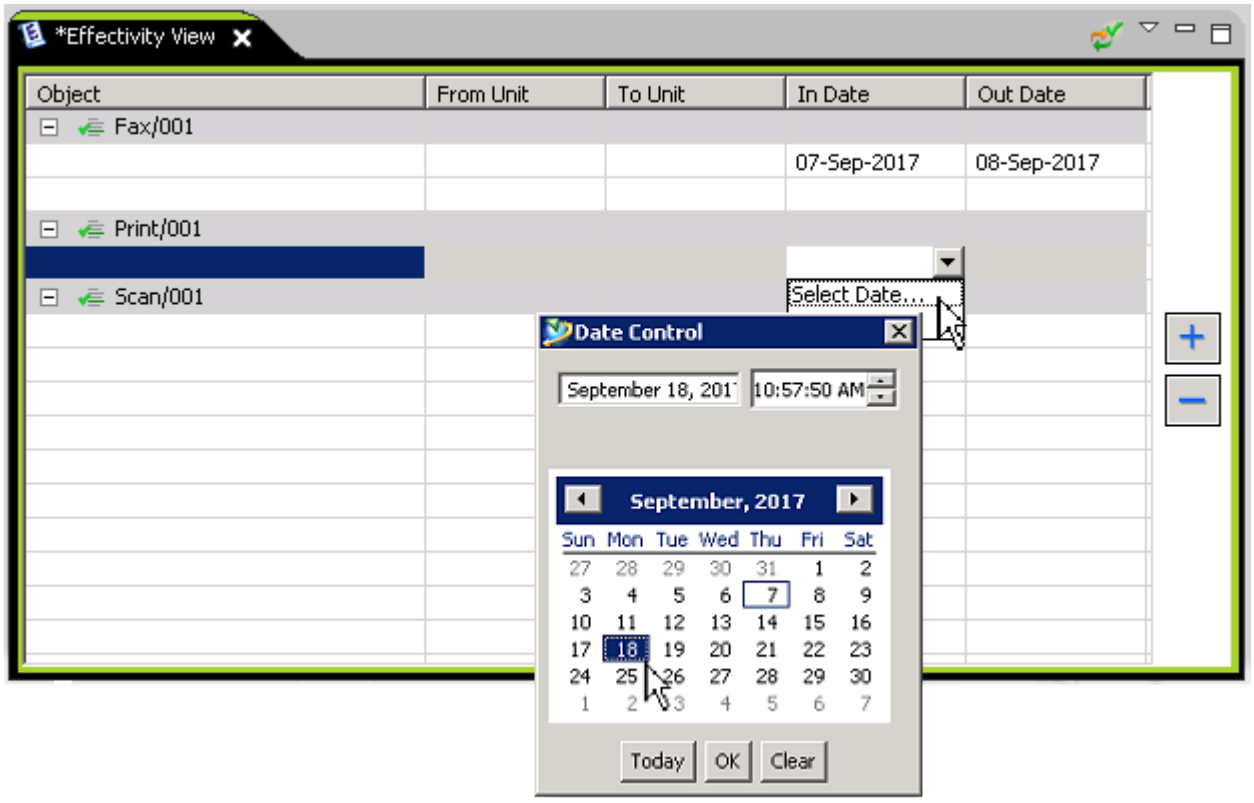
   You can also double-click the effectivity cell for the corresponding feature revision in the **Variability Explorer** view title bar.

   > Note:
   >
   > The **PCA_effectivity_shown_columns** preference controls which columns are visible in the **Effectivity View** dialog box.

   Teamcenter displays the **Effectivity View** dialog box.

3. Click + to add a new blank effectivity expression (row).

4. Click the ▼ in the corresponding cell to select a range of dates or unit numbers for which the object is effective. You can also specify a combination of unit numbers and dates.



Note:

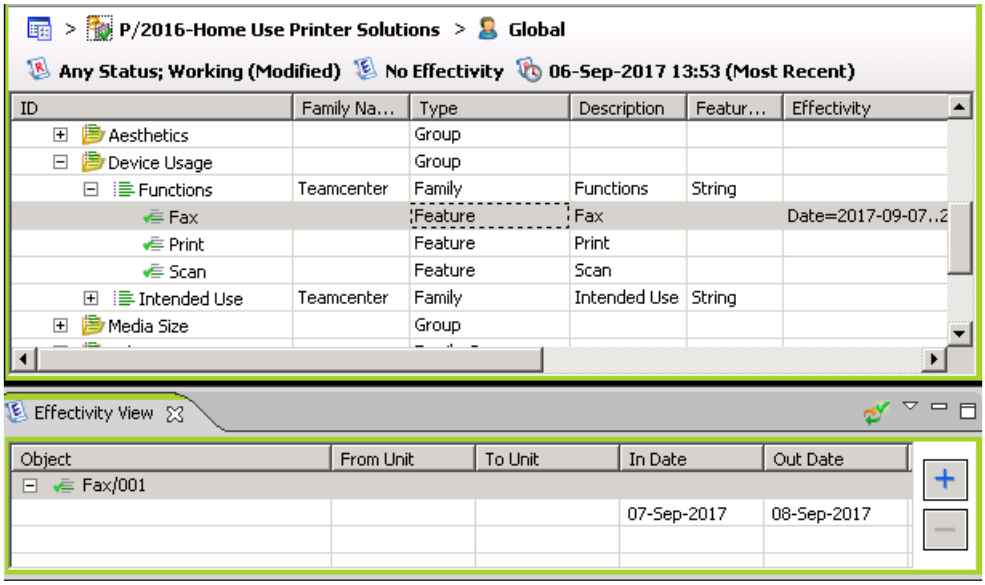
Type or choose a range of dates, using the **Date Control** window, for which the object is effective.

If you enter dates and units in the same row, Teamcenter interprets this as an AND condition, for example, “Include units 5-6 AND effective between dates 7/1/2015 to 8/1/2015”. If you enter dates, units or both on different rows, Teamcenter interprets this as an OR condition, for example, “Include units 5-6 OR effective between dates 7/1/2015 to 8/1/2015”.

You can specify open–ended effectivity to infinity (click **UP**) or until stock out (click **SO**).

5. Click **Save** on the main toolbar or **Save** in **Effectivity View**.

   Teamcenter saves and applies the effectivity you defined.



8. Define variants and associate a configurator context with a structure

Define variants

Variants are of two types, variant rules or variant criteria.

Your administrator controls if you create variant rules or variant criteria using the **Cfg0CreateVariantRuleType** preference. The preference can be set to either **VariantRule** or **Cfg0VariantCriteria** which allows you to create either variant rules or variant criteria, respectively. By default, it is set to **Cfg0VariantCriteria**. Siemens Digital Industries Software does not recommend using both variant criteria and variant rules at the same time. If you have started using variant criteria, do not switch back to variant rules.

You can author variant rules or variant criteria for selecting features in a configurator context. These selection rules include defaults and derived features. In addition to selected features, variant rules store user session information. A variant rule session information consists of severity, configuration mode, revision rule, effectivity, and a rule date saved along with an expression. A variant criteria object is a revisable subtype of a variant rule. You can release, revise, and copy your variant criteria. If your business process requires you to manage the lifecycle of a variant rule, such as its effectivity, revision rule, and rule date, then it is recommended to use variant criteria.

Create a variant rule or a variant criteria

You can create a variant rule or variant criteria to select features in a configurator context.

Your administrator controls if you create variant rules or variant criteria using the **Cfg0CreateVariantRuleType** preference. The preference can be set to either **VariantRule** or **Cfg0VariantCriteria** which allows you to create either variant rules or variant criteria, respectively. By default, it is set to **Cfg0VariantCriteria**.

The following example shows how to create a variant rule in the **Saved Variant Rules** view. You create variant criteria in a similar way.

You can also save your custom configurations as saved variant rules or variant criteria when defining your configurations in the **Variant Configuration** view by clicking  . The system saves your selections along with the session information.

Procedure

1. Open the configurator context and then make the **Variability Explorer** view active.

2. Click  to add a new saved variant rule (SVR).

Teamcenter displays the **Saved Variant Rules** view.

3. Type a name and description of the rule and press Enter.

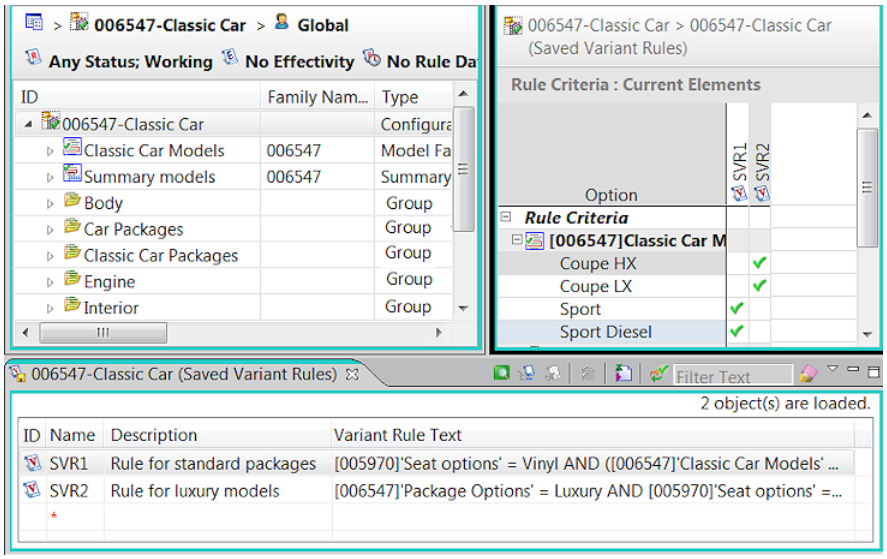
   While creating a new variant rule or variant criteria, click the **View** menu and choose the **Configuration Profile Settings** menu to select the profile settings from the **Configuration Profile Settings** dialog. The selected values are saved as configuration session information on the newly created variant rule or variant criteria.

4. Click **Save** on the main toolbar to save a newly created SVR.

5. Click to show all rules, if any, or **Show selected rules** to only view the selected rules.

   Teamcenter displays the **Variant Expression Editor** view.

6. Construct a variant expression for the selected rule in Teamcenter, as shown in the following example:



- Features are represented by columns, organized by their families. You specify an expression by clicking the cells below the required features one or more times.
- Selections are indicated by a check mark or crossed circle in the cell.
- Selections in the same column are combined (AND function), except for selections in the same family, which are alternatives (OR function). You can add multiple rows for the same expression.
- Multiple subexpressions are combined (OR function).

> Warning:
>
> Do not create configurator rules, variant conditions, or saved variant rules by combining a product model with a summary model or a product line.

You can also edit an existing rule by changing any of the fields. Updated rules are displayed but not yet saved.

7. You can split the SVR in the **Variant Expression Editor** view by right-clicking the column header and choosing **Split Expression** .

   Teamcenter creates another column along with the selected column and allows you to author a new expression. The newly created column is connected using a logical OR operation with the original expression column.

   Teamcenter validates an entire expression or each column separately as an individual configuration and provides feedback.

8. Repeat the previous steps as necessary for other rules.

9. Click **Save** on the main toolbar to save all additions, deletions, and edits to the database.

   When you create a variant rule from the **Saved Variant Rules** view by authoring in **Variant Expression Editor**, then the system saves the expressions for the variant rule. Configuration session information is saved on a new variant rule as per the selections in **Configuration Profile Settings** dialog.

   > Note:
   >
   > The variant expression editor is used for creating variant rules. Do not use it to update an existing SVR when it has been already created or updated using the **Variant Configuration** view. The SVRs created or modified using the **Variant Configuration** view has both the user expression and the expanded expression. The variant expression editor does not support the editing of both the user expression and the expanded expression in one view.

10. If Teamcenter detects any violations of configurator rules, the violation message appears.

    The rule is displayed with a symbol and a tooltip with a detailed violation message.

11. To find variant criteria or variant rules based on expressions:

    a. Select the expressions in the **Variant Configuration** view.

    b. Click **Search Variant Rules based on Expression** in the **Saved Variant Rules** view.

The system displays results using the rule date and the session information set in the **Variability Explorer** view.

To clear the search, click **View menu** ▾ again and select **Clear Search Results**. Similarly, you clear your selections in the **Variant Configuration** view and click 🔍 to clear the search results. Then, repeat searching for SVRs or variant criteria based on your new selections.

12. To view multiple revisions of variant criteria in the **Saved Variant Rules** view, click **View menu** ▾ and then select **Show All Revisions**.

    The system displays all available revisions irrespective of the session information set in the **Variability Explorer** view.

    Click **View menu** ▾ again and deselect **Show All Revisions** to return to displaying only the applicable revision based on the session information set in the **Variability Explorer** view.

    > Tip:
    >
    > Multiselect multiple revisions of variant criteria and open them in the **Variant Expression Editor** to compare.

13. In the **Variant Expression Editor** or the **Saved Variants Rules** view, select one or more SVRs and click **Validate Configurator Rules** to validate SVRs on demand.

    In the **Saved Variants Rules** view, if the validation check failed for an SVR, the rule is shown in red and the error message appears. You can also view a tooltip with a violation message if you hover the cursor over the failed SVR rule.

Submit variant rules and variant criteria to the workflow

You can submit variant rules and variant criteria to the workflow.

Procedure

1. Open the configurator context and then make the **Variability Explorer** view active.

2. Click to open the **Saved Variants Rule** view.

3. Select the variant rule or variant criteria you want to submit to the workflow.

   Your administrator controls if you create variant rules or variant criteria using the **Cfg0CreateVariantRuleType** preference. The preference can be set to either **VariantRule** or **Cfg0VariantCriteria** which allows you to create either variant rules or variant criteria, respectively. By default, it is set to **Cfg0VariantCriteria**.

4. Choose **File→New→Workflow Process**.

5. Select the appropriate template, for example, **TCM Release Process**, and specify other values as appropriate for the required fields, and click **OK**.

Revise variant criteria submitted to the workflow

You can revise variant criteria submitted to the workflow.

Procedure

1. Open the configurator context and then make the **Variability Explorer** view active.
2. Click  to open the **Saved Variants Rule** view.
3. Select a variant criteria that has been submitted to the workflow.

   The **Release Status** box displays the release status.
4. Choose **File→Revise** and click **Yes**.
5. To see the revision, change the revision rule in the header of the configurator context in the **Variability Explorer** view to **Working; Any Status**.

Change the rule date behavior for variants rules or variant criteria while loading or saving them

Starting Teamcenter 11.6 release, the variant rules and variant criteria objects were enhanced to store revision rule, effectivity, and rule date information from the user session in which it was created or modified. This stored information is used when the variant rules and variant criteria is used for BOM solve or when it is loaded in views such as **Variant Configuration**.

Different business scenarios demand more flexibility around the rule date stored on variant rules and variant criteria.

As an example, if the configurator data goes through daily modification, then the variant rules or variant criteria created in the past with an older rule date cannot provide BOM solve results as per the modified data. In this scenario, users expect the rule date as either the System Default or the Latest or NULL.

- Saving variant rules or variant criteria in the **Variant Configuration** view.

  When the variant rule and variant criteria are saved from the **Variant Configuration** view, the rule date set on variant rules or variant criteria is the rule date from this view.
- Saving variant rules or variant criteria in the **Saved Variant Rules** view.

When variant rules or variant criteria are saved from the **Saved Variant Rules** view, the rule date set on variant rules or variant criteria is the rule date from the **Explorer** view.

The following rule date behavior is applied while loading variant rules and variant criteria in the **Variant Configuration** view:

Rule Date Set on Variant Rule	Description	Variant rules or variant criteria created on 20 July-2021 at specified system time	Rule Date evaluated when a variant rules or variant criteria is loaded on 21-July-2021
System Default	Ignores the rule date on the variant rule or variant criteria and treats the rule date as the system default.	20-July-2021 15:00:00	21-July-2021 00:00:00 (System Default)
Latest	Ignores the rule date on the variant rule or variant criteria and treats the rule date as the current time on the clock.	**20-July-2021 10:00:00**	**21-July-2021 11:11:11** *Time on clock* **(Latest)**
20-July-2021 10:00	Does not ignore the rule date on the variant rule or variant criteria and uses whatever rule date is available on the variant rule.	**20-July-2021 11:00:00**	**20-July-2021 10:00:00** (Uses the rule date on which the variant rule or variant criteria was created.)
No Rule Date	Ignores the rule date on the variant rule or variant criteria and treats the rule date as null.	**20-July-2021 10:00:00**	**No Rule Date**

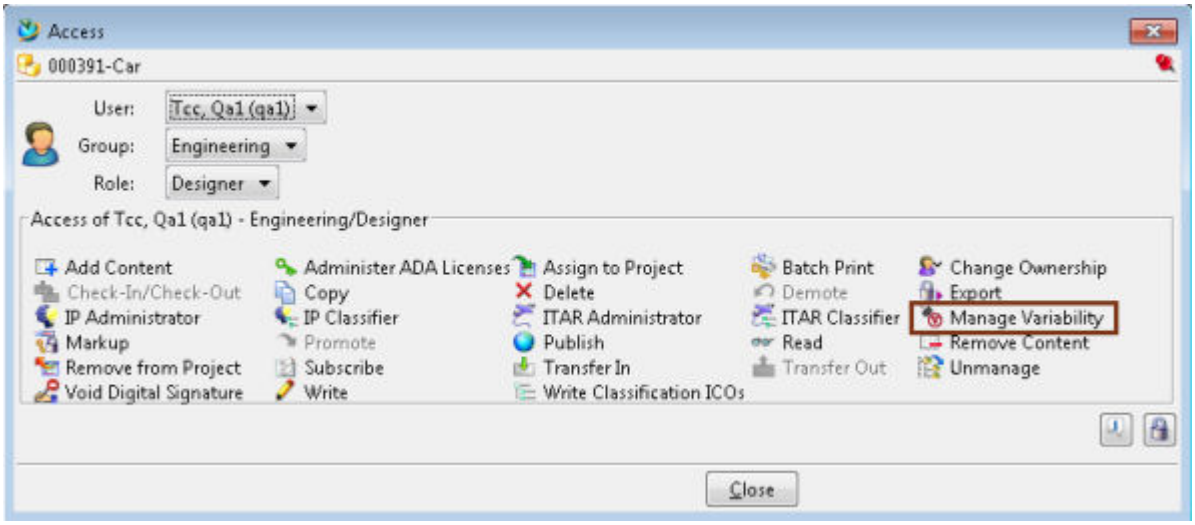
Associate a configurator context with a structure

To obtain the required variability data such as *families*, *features*, and *rules* from a configurator context, you associate the configurator with the product structure.

You can perform this association only if you have the **Manage Variability** privileges. To verify that you do:

1. In My Teamcenter, right-click the topmost item of the product structure and click **Access**.

2. In the **Access** dialog box, verify that the **Manage Variability** privilege is enabled.



Note:

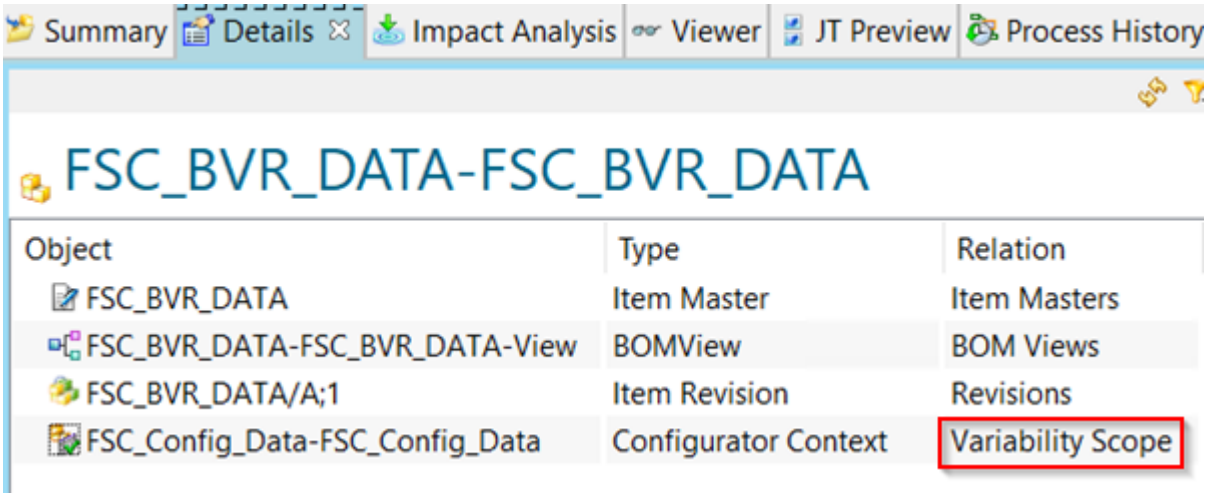
Using a mixture of legacy and product configurator variants can cause data corruption. By default, the system uses the preference **ME_EnableMixVariantModelCheck** to prevent loading structures that use a mixture of these variants. If you are using only one variant, or no workflow results in mixed variants, change the value of **ME_EnableMixVariantModelCheck** to **false**. This value enables the system to open structures without a check.

Associate a configurator context with a structure

1. In My Teamcenter, search for the structure and the configurator context.

2. Right-click the product structure and choose **Paste**. You cannot associate the configurator context with the item revision of the structure.

   To verify if the configurator context is associated with the structure, select the product structure and click the **Details** view. The **Relation** column must show **Variability Scope**.



Only the configurator context associated with the topmost line item of the structure is considered for authoring variant conditions.

> Tip:
>
> If you are likely to use this configurator context as the basis for several structures, set this as the default context.

9. Configure and analyze

Configure with custom variant configuration (manual validation)

Manual configuration mode is primarily used to perform a study or an overlay analysis in the application that manages product content. Manual configuration is also used by configuration analysts to study the system response based on variety of selection paths. Using this mode, you can build and save valid configurations or study configurations.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variant Configuration** view displays **Name** instead of **ID** in the corresponding areas.

1. Set a revision rule, effectivity, and rule date by clicking the corresponding hyperlink.

   If an unconfigured feature exists in a configuration, such as after applying a saved variant rule, the system displays a question mark symbol next to the selected feature.

2. Select manual validation mode by clicking **Guided configuration mode** off. (The configuration mode always defaults to manual when you open a new session.)

   By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** configuration mode with all the available variant features including standalone features, feature packages, feature summaries, and summary models. You can select more than one feature from the single select family in the **Overlay** mode.

   To switch to the **Order** configuration mode, click **Overlay Configuration Mode**. When you switch between the **Overlay** and **Order** modes, Teamcenter resets the system selections that were derived based on rules prior to switching the mode.

   If you hover the cursor over the button in the order mode, the tooltip displays **Order Configuration Mode**.

   Your administrator can change the default configuration mode using the **Cfg0DefaultValidationMode** preference. By default, it is set to **Order** mode.

   Observe that the header of the view displays the completeness check indicators. The icons appear active or inactive based on the results of the validation.

   - Valid and complete product configuration .

     Valid and complete means that the BOM is targeting a complete structure, that is, a 100% BOM. When a configuration is valid and complete, you could take this configuration input and execute an order entry in the order system.

On the contrary, if you do not see this completeness icon, you should not typically perform the above action. It means that you have less content or more content.

When a configuration is valid and complete, it assigns a value to every family. This is irrespective of whether the family is mandatory or discretionary.

When you click **Validate**, the system only validates your input.

- indicates that your input expands into a valid and complete configuration if you click **Expand**, such as it is a prediction of what would happen if you choose to expand your configuration.

When you click **Expand**, it means that the expanded configuration in the current configurator context is complete.

- indicates that your input was valid and was successfully expanded into a valid and complete configuration. If the icon was not active prior to clicking **Expand**, the **Variant Configuration view** was changed during the **Expand** operation.

- Valid and incomplete product configuration .

  When you click **Validate**, it means that the input criteria is valid during the validation.

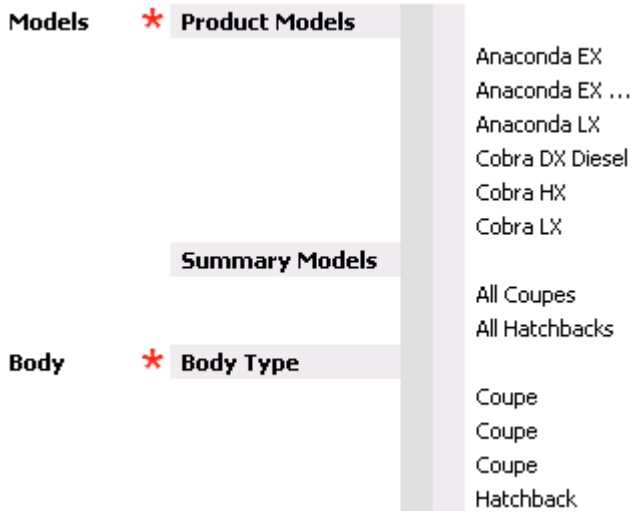
  When you click **Expand**, it means that the expanded expression is incomplete. For example, mandatory families may need to be selected to complete this configuration. You can also view a tooltip with a violation message with a list of incomplete families if you hover the cursor over the result. The tooltip displays a list of incomplete mandatory families.

- Invalid product configuration .

  It means that the input criteria is invalid based on the configurator rules set in the current configurator context.

> Note:
>
> Contact your system administrator if the system returns a validation failure without showing a list of information, warnings, or errors. It may be an access control issue.

A red asterisk next to the family indicates that a variant feature selection is required. Once you make a selection that completes a required configuration, the red asterisk disappears.



> Warning:
>
> Do not create configurator rules, variant conditions, or saved variant rules by combining a product model with a summary model or a product line.

3. Click the cell to select the variant condition you want to use to define the configuration:

   - Click one time to display a green check mark ✔ to include the variant condition when defining the configuration.
   - Click two times to display a red circle backslash ⃠ to exclude the variant condition when defining the configuration.
   - Click three times to display a blank cell to indicate the variant condition is not used when defining the configuration.

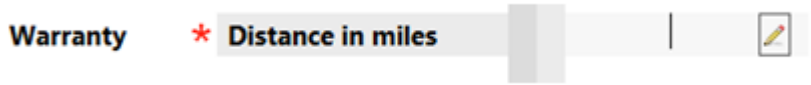
   If the cell is positively selected with a green check mark ✔ , it means **ANY**. If it is negatively selected with a red circle backslash ⃠, it means **NONE**.

   Multiple variant conditions are connected using a logical **AND** operation in the variant expression that defines the configuration.

4. If you want to override the system-assigned default feature, click the system default feature once to deselect, and then click another feature from the same family.

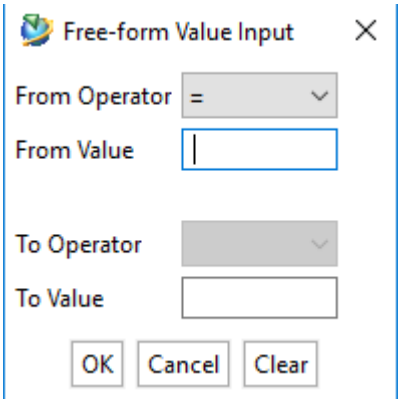
5. If the feature is a free-form text, perform the following steps:

   a. Click to the right of the cell under the corresponding feature to display the edit pencil symbol.



b. Click to type the values.

c. In the **Free-form Value Input** dialog box, type the values to set the variant condition.

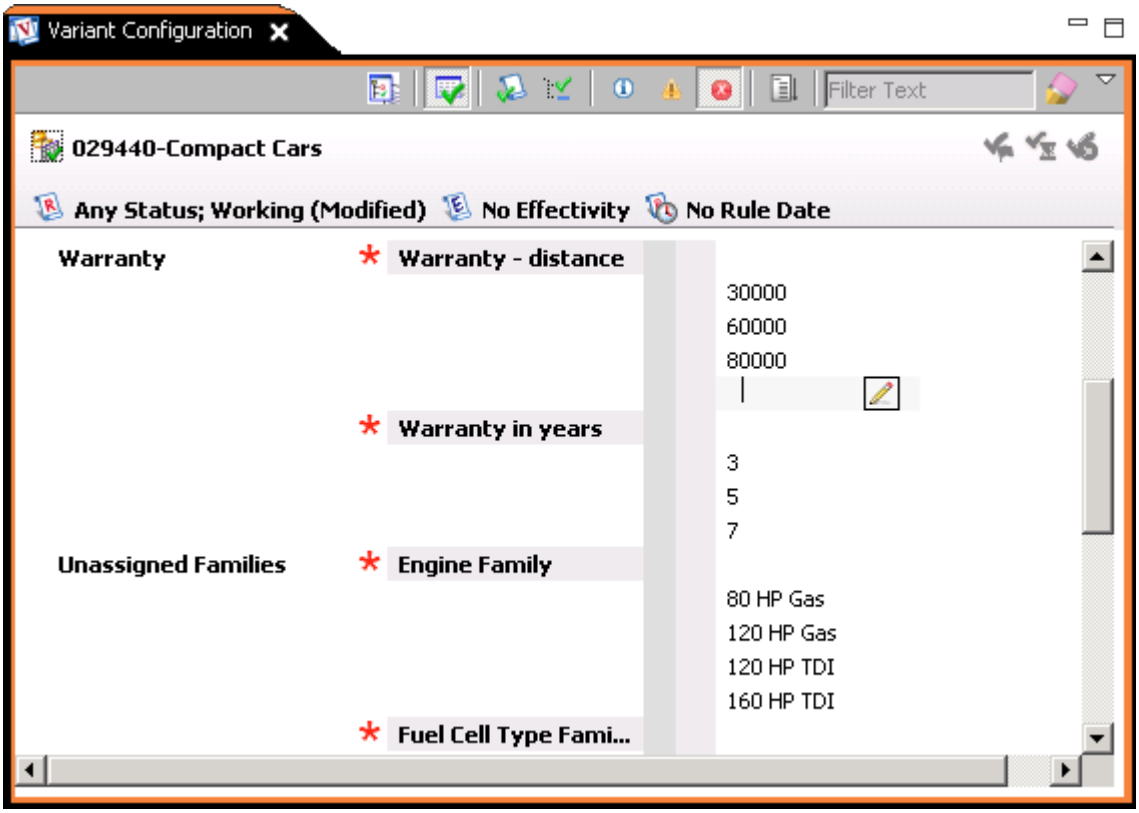


d. Click **OK**.

If the entered value contains any validation or formatting issues, the free-form feature is displayed in red in the feature text field and flagged with an error message.

6. Use the above procedure to specify values for free-form non-string, numerical, or date type families.

Click below the last cell with enumerated values to display a new line and the edit pencil symbol.

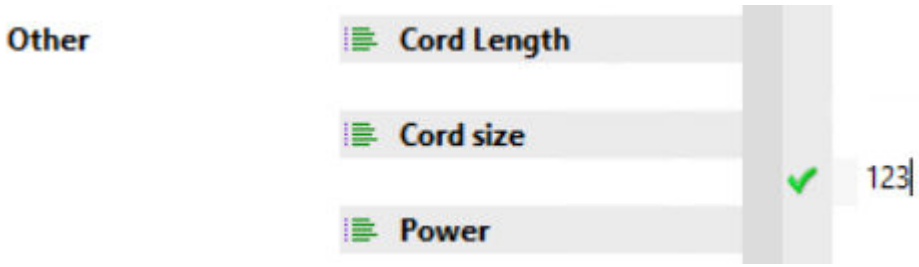


You can use operators only for the enumerated values for numeric families.

7. If a family is defined as free-form string, perform the following steps:

   a. Click to the right of the corresponding cell under the free-form string family.

   b. Enter values directly in the cell.

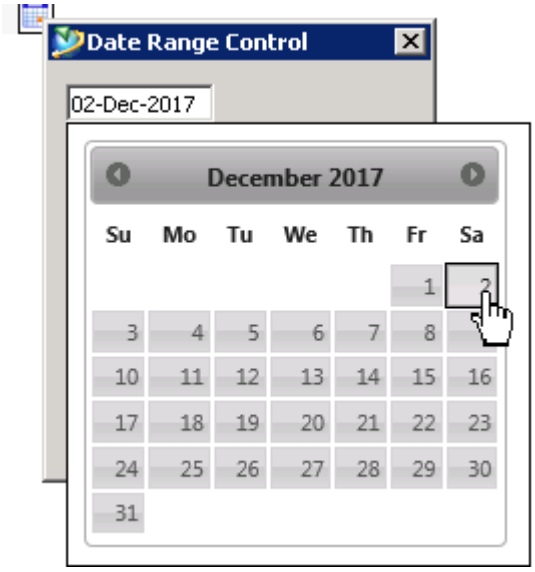


8. If a family is defined as date, perform the following steps:

   a. Click to the right of the cell to display the date symbol.

b. Click to define dates.

c. In the **Date Range Control** dialog box, click today's date to open the calendar to set the required date or type the values and ranges.



The calendar initially displays the current day, month, and year.

For a non-free-form data type, predefined features are not listed in the drop-down list when choosing a range.

For example, if an integer family **Horse Power** has features of **200**, **400**, and **500**, then the dialog box displays **200**, **400**, and **500** in the drop-down list. If a family **Release** has features of **01-01-2018** and **01-01-2019**, then the dialog box does not display a drop-down list with these features. If you want to use a range by using these features, select the precise date using date control in the **Date Range Control** dialog box.

d. Click **OK** to set the date.

9. If a family is defined as a dynamic family, when you select a standalone feature in the manual mode, it is valid across all the selections in the dynamic families where this feature resides.

Your administrator can enable the creation of dynamic families and standalone features using the **Cfg0EnableDynamicFamilySupport** preference.

> Note:
>
> The standalone features and the dynamic family concept are in an incubation phase. They should only be used with the approval from Siemens product management. They require authoring in Active Workspace and consumption of configurator data.

10. If you do not want to apply constraints, deselect **Apply constraints**. By default, this option is selected.

    - If selected, constraints are applied while applying the variant rule expression on the content.
    - If deselected, only user selections are applied.

11. If you do not want validation rules to participate during criteria expansion, deselect **Allow validation rules to expand**. By default, this option is selected.

    - If selected, allows validation rules to participate during criteria expansion.
    - If deselected, only violations, if present, are reported. The system selections based on validation rules does not occur in the expanded criteria.

    For more information, see **Example: Apply constraints and allow validation rules to expand**.

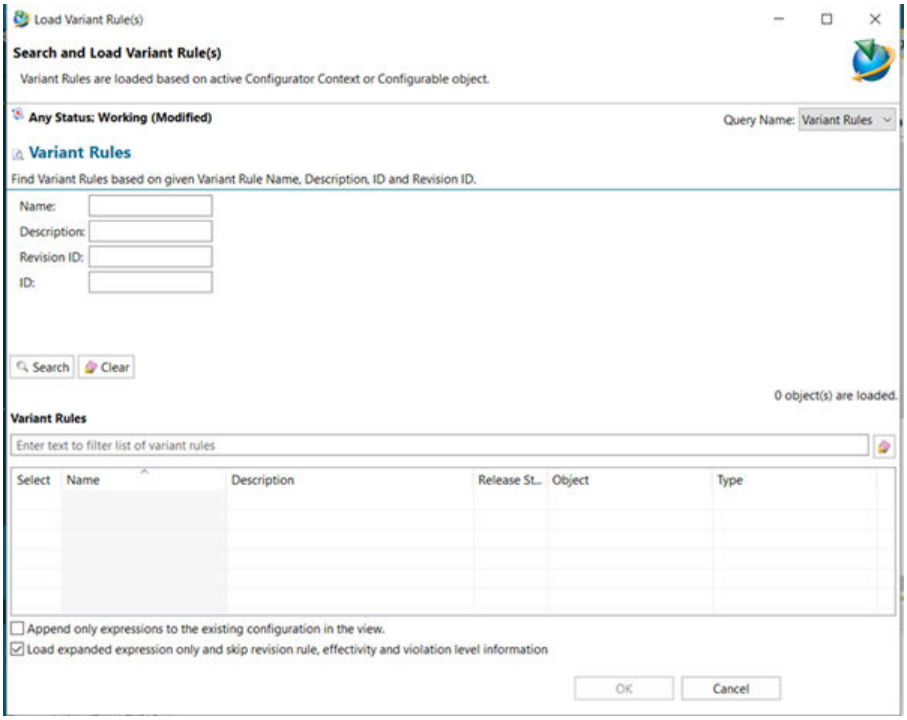
12. To remove all variant expressions, click **View menu** and choose **Clear Expression**.

    The system re-evaluates the configuration and displays features that were hidden based on the previous selections.

13. To load an existing configuration for viewing or editing, perform the following steps:

    a. Click **Load Variant Rule(s)** .

       Teamcenter displays the **Load Variant Rule(s)** dialog box. You can search and filter variant rules or variant criteria using your defined criteria in this dialog box.



b. Type a name or description as your search criteria to find variant rules or variant criteria.

- For variant criteria only, you can also search using the revision ID or ID. Additionally, you can specify the revision rule and the rule date in the **Load Variant Rule(s)** dialog box to locate the required revision of your variant criteria.

Press Enter or click **Search**.

Your administrator controls which queries and search criteria display in the **Load Variant Rule(s)** dialog box using the **PCA_VariantRuleSearch_SavedQueries** preference. This preference sets the default value as **Variant Rules** which is a generic saved query used to perform a search for variant rules. You can define your own saved query for variant rules and add its name to the **PCA_VariantRuleSearch_SavedQueries** preference.

If your administrator adds configurator context to the search criteria, the system locates only variant rules that are shared with the current and active configurator context in the **Variant Rule(s)** view.

Your administrator controls if you create variant rules or variant criteria using the **Cfg0CreateVariantRuleType** preference. The preference can be set to either **VariantRule** or **Cfg0VariantCriteria** which allows you to create either variant rules or variant criteria, respectively. By default, it is set to **Cfg0VariantCriteria**. If the value is not set, you see an error message notifying you that no valid variant rule is found in the system.

All variant rules that match your search criteria are listed in a table at the bottom of the dialog box. Additionally, it displays the number of variant rules found in your search.

c. If needed, you can filter the results by typing your criteria in the filter box.

d. Check one or more boxes to select product configurations you want to load and click **OK** to replace the existing selections. By default, **Load expanded expression only and skip revision, effectivity and violation level information** is selected.

> Note:
>
> The default state of the check box is controlled by the **Cfg0LoadExpressionInfoForVR** user preference. By default, it is set to **false** and the variant rule is loaded without session information.

The system loads expressions of the selected variant rules or variant criteria in the **Variant Configuration** view.

If not selected, the configuration session information with expressions is loaded for the selected variant rule. A variant rule session information consists of severity, validation mode, revision rule, effectivity, and a rule date saved along with an expression.

When you load the rule without the session information, the system displays it as **Custom configuration**. If you modify and save it, the system saves it as a new rule.

If you load the entire rule including all of the expressions and the session information, the system displays the name of the variant rule or the ID, revision, and the name of the variant criteria. If you make any changes to that rule, the system overrides the information and saves the changes in the same rule. Validation information is not saved along with the rule. It is recommended to validate the rule after you reload it.

When you make changes to variant rules or variant criteria, such as make a new selection, change a revision rule or effectivity, the system resets the validation flags.

Alternatively, you can select **Append only expressions to the existing configuration in the view** and Teamcenter loads each in a separate column, preserving the existing selections. Teamcenter loads only saved expressions. The system applies the configuration to the current content and logically overlays multiple columns, resulting in a 120% BOM configuration or an imprecise configuration.

When using the **Load Variant Rule(s)** dialog box to append new saved variant rules to your configuration using **Append only expressions to the existing configuration in the view**, you can only view the already loaded SVRs but you cannot deselect them. You can choose to load other available saved variant rules.

When you overlay configurations, the system displays each configuration in a separate column during authoring and validation. Configurations are also expanded column by column. Number of columns in the input and output configurations are maintained by the system.

If you select variant rules to load with **Append only expressions to the existing configuration in the view** not selected, then the system does not append the columns for the selected rules. If you search and load variant rules or variant criteria multiple times, then the system constructs results only for the newly selected objects and does not include the already loaded ones.

> Note:
>
> If you make imprecise selections, you cannot switch to the guided (active) mode. The system generates an error message stating that the active validation mode only supports the precise configuration.

14. (Optional) Click **View menu** and choose **Clear Variant Rule** to unload the variant rule or variant criteria from the view and reset the session information, including severity, validation mode, revision rule, effectivity, and the rule date, to the default state.

15. (Optional) Click **View menu** and choose **Clear Un-configured Feature** to remove unconfigured features.

    The expression on the unconfigured feature in the view is removed. Also, the row holding the unconfigured feature is removed.

16. Click **Validate**. The system validates your input and considers all system rules to validate and report the verdict. It does not change the input expression. It does not consider rules below the selected severity level, such as default rules.

    You can specify operators for numerical or date family when validating your configuration.

    Teamcenter provides feedback after performing a set of functions as follows:

    a. Validates the selected criteria against the system rules with the severity equal or greater than the input severity.

    b. Applies all configurator rules.

    c. Reports violations based your selections.

    If prior to validation, all families have precise selections, the system displays the **Valid and complete product configuration** status. Otherwise, the system displays the **Valid and incomplete product configuration** status and provides additional information indicating the reason for the invalid status.

d. Reports violation on the free from families where value specified is outside their minimum or maximum range.

If you hover over the violation, the tooltip displays object details that caused this violation, such as specific features or configurator rules. If needed, you can select and copy the violation message.

If your configuration contains an inconsistent or conflicting rule set, then the violation occurs irrespective of any selection in the **Variant Configuration** view or for any other validation. In such cases, the violation may not be displayed on your selection. For example, if no features from mandatory families are available, then during the validation, the violation occurs. The violation is displayed on features of mandatory families even if they are not a part of the input configuration to validate.

Configurator constraint rules, such as inclusion or exclusion rules, specify a severity level. When you are validating your configuration, the system reports the constraint rule violations with the specified severity level.

17. You can perform the following actions to modify the interaction for validations during the manual configuration:

    a. Click **Set Info level violations to be fetched while applying validations** to switch between displaying and enforcing or hiding and dropping of all informational messages.

    b. Click the **Set Warning level violations to be fetched while applying validations** to switch between displaying and enforcing or hiding and dropping of all warning messages.

    Error level violations are always displayed.

    During the validation process, if you have multiple rules with conflicting severity levels, the systems validates and enforces the rules based on the following severity ranking:

    a. Error (highest severity)

    b. Warning

    c. Information

    d. Package Default

    e. Default

    f. System Enforced Defaults (lowest severity)

18. Click **Expand**. The system considers all system rules to validate your input and assigns features or precise values to the feature families using system knowledge.

If **Allow Validation Rules to Expand** button is deselected during expansion, only violations are reported, if present. The system selection based on validation rules does not occur in the expanded criteria. By default, Teamcenter displays the **Variant Configuration** view with this button as selected.

- In the **Overlay** mode, the system assigns values based on the valid input criteria.
  - The system appends your input in the **Variant Configuration** view with the values that the system identified as the only possible solution in the selected families. The system considers all configurator rules with the severity equal or greater than the input severity.

    Note:

    The families with multiple possible solutions are left unassigned.

  - The system informs you if the expanded criteria is complete or incomplete.
- In the **Order** mode, the system assigns values based on the valid input criteria, similar to the **Overlay** mode. Moreover, it provides additional values to help you create 100% configuration or order configuration. These additional values are provided for all families.

  The system suggests the empty value as the default if there are multiple solutions or for an optional family. This means the resulting configuration is not choosing any values for the optional family.

  All of the nonconflicting low severity rules, including default rules, are considered.

19. Click to save the custom rule.

    If you started from a blank view, Teamcenter displays the **Save Product Configuration** dialog box; otherwise it saves the changes to the existing product configuration.

    If you exit the **Variant Configuration** view without saving, you are prompted to save the changes.

20. To save a new configuration, in the **Save Product Configuration** dialog box, type a name and description, and click **OK**.

    Teamcenter creates a custom configuration with the specified name and variant configuration. The system adds the name of the saved variant rule to the title bar.

21. To clear a previously loaded custom configuration, click **View menu** and choose **Clear Variant Rule**.

    The system removes the name of the saved configuration from the title bar and then resets user and system selections to the initial state.

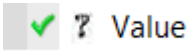
Configure with guided variant configuration (active validation)

Guided configuration mode is primarily used to simulate the configuration behavior for customer interactions. The users of this configuration want to be sure that there is a valid configuration path for every planned product variant. Similar to manual configuration mode, validated configurations can be saved and used to track primary configurations.

If your system is configured with the global constant that points to **Name** as a primary business property, the **Variant Configuration** view displays **Name** instead of **ID** in the corresponding areas.

1. Set a revision rule, effectivity, and rule date by clicking the corresponding hyperlink.

   If an unconfigured feature exists in a configuration, such as after applying a saved variant rule, the system displays a question mark symbol in front of the selected feature.

2. Click to use guided validation by clicking **Guided configuration mode** on. (The validation mode always defaults to manual when you open a new session.)

   By default, Teamcenter displays the **Variant Configuration** view in the **Overlay** configuration mode.

   To switch to the **Order** configuration mode, click **Overlay Configuration Mode**. When you switch between the **Overlay** and **Order** modes, Teamcenter resets the system selections that were derived based on rules prior to switching the mode.

   If you hover the cursor over the button in the order mode, the tooltip displays **Order Configuration Mode**.

   Your administrator can change the default configuration mode using the **Cfg0DefaultValidationMode** preference. By default, it is set to **Order** mode.

   Observe that the header of the view displays the completeness check indicators. The icons appear active or inactive based on the results of the validation.

   - Valid and complete product configuration

     Valid and complete means that the BOM is targeting a complete structure, that is, a 100% BOM. When a configuration is valid and complete, you could take this configuration input and execute an order entry in the order system.

     On the contrary, if you do not see this completeness icon, you should not typically do any of the above actions. It means that you have less content or more content.

When a configuration is valid and complete, it assigns a value to every family. This is irrespective of whether the family is mandatory or discretionary.

- Valid and incomplete product configuration 

  In the guided mode, Teamcenter does not provide a list of incomplete mandatory families.

  > Tip:
  >
  > To locate a list of missing required families, switch to the manual mode. For the valid and incomplete product configuration  in the manual mode, Teamcenter generates a tooltip with a list of incomplete mandatory families.

- Invalid product configuration 

Teamcenter shows a list of groups with the first in bold and arrows above and below the list. Any features that violate constraints are indicated. The families and features of the group are displayed in the right pane.

A red asterisk * next to the family indicates that a variant feature is required. Once you make a selection, * next to the family disappears.

- A  symbol is displayed next to the system-assigned available features.

  It indicates that the displayed features are enforced in the current configuration. Typically, features configured due to include, exclude, and availability rules are shown as system-assigned.

  A system selection is only shown for families with exactly one solution.

- A  symbol indicates that the presence of the selected feature is denied by the system due to configurator constraints, such as inclusion, exclusion, or availability rules.

- A  symbol is displayed next to the default features when the selection is enforced by:

  - A default rule.

    The applicability of the default rule is satisfied either directly by a user–defined criteria or by a system selection enforced by include, exclude, and availability rules.

  - An include rule.

    The applicability of the include rule is satisfied through a default rule.

- A  symbol indicates that a selected feature is denied by the system due to the default rule.

If you have an imprecise configuration displayed using manual validation, you cannot switch to the guided (active) mode. The system generates an error message stating that the active validation mode only supports the precise configuration.

Guided configuration displays variant data within one group at a time. These are user-created groups. Every configurator context contains a default **Unassigned Families** group. As this group is system-generated and not manually created by a user, it is *not* displayed in a guided configuration mode. Thus, the variant data within the **Unassigned Families** group is also *not* displayed in a guided mode. However, in a manual mode, both the **Unassigned Families** group and its variant data are displayed. Families and features become visible in a guided configuration mode, once you create a new parent group or move the unassigned families with the corresponding features to an existing group.

Standalone features and dynamic families with standalone features are also displayed, if available.

The system displays all of the available Boolean features and features from the multi-select families, only if they are made explicitly available.

> Note:
>
> Product lines, summaries, summary models, and families that reside in the **Unassigned Families** group are not displayed during the guided validation.

3. Click the cell to select the variant condition you want to use to define the configuration.

   - Click one time to display a green check mark ✔ to include the variant condition when defining the configuration.
   - Click two times to display a blank cell to indicate the variant condition is not used when defining the configuration.
   - Click the , , or system-assigned features once to change them to the user-assigned selections.

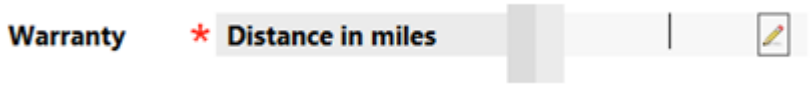
   If the cell is positively selected with a green check mark ✔ , it means **ANY**. If it is negatively selected with a red circle backslash ⃠, it means **NONE**.

   Multiple variant conditions are connected using a logical **AND** operation in the variant expression that defines the configuration.

   If you select features that define a model, validation automatically selects a model. When you select a product model or a feature package, then the system adds the corresponding product model or feature package members, respectively, to the configuration.

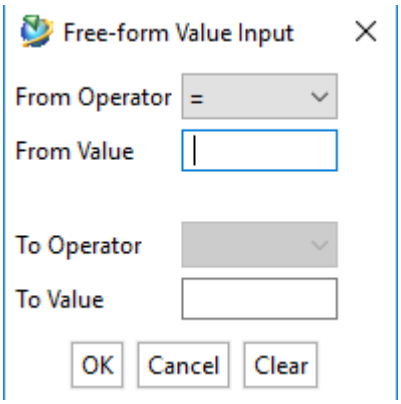
4. If the feature is a free-form text, perform the following steps to set the feature:

a. Click to the right of the cell under the corresponding feature to display the edit pencil symbol.



b. Click to type the values.

c. In the **Free-form Value Input** dialog box, type the values to set the variant condition.



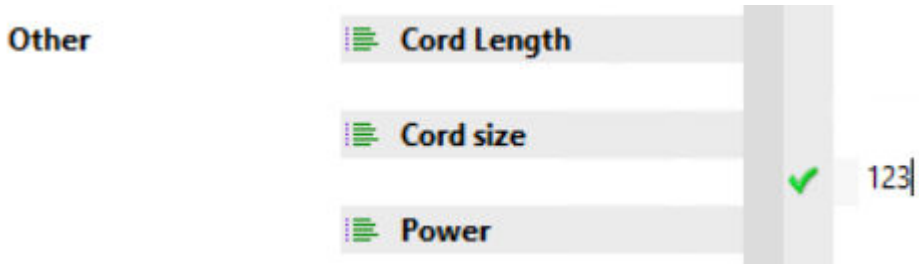
d. Click **OK**.

If the entered value contains any validation or formatting issues, the free-form feature is displayed in red in the feature text field and flagged with an error message.

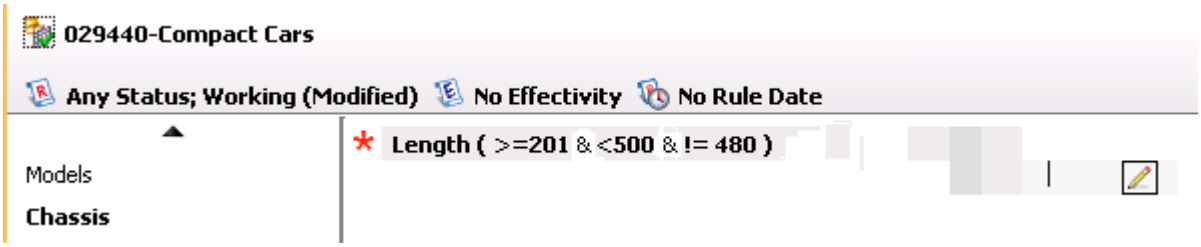
5. If a family is defined as a free-form string, perform the following steps:

a. Click to the right of the corresponding cell under the free-form string family.

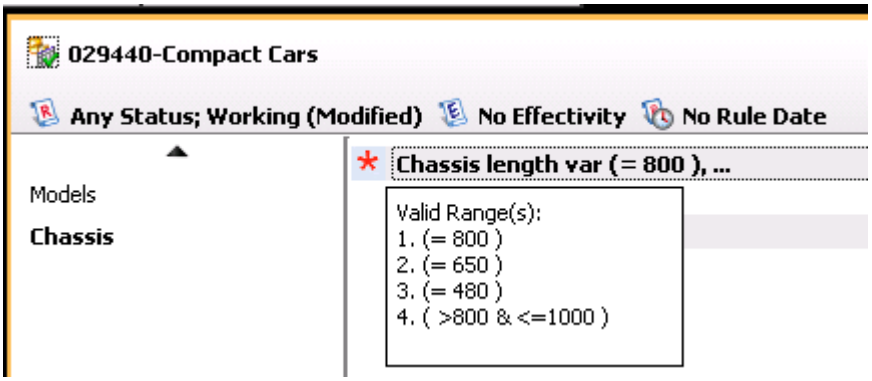
b. Enter values directly in the cell.



6. During guided validation mode, for enumerated and free-form non-string families, the system displays allowable ranges by considering all of the configurator rules.



The system displays the updated range and considers all possible subranges. The tooltip displays the disjointed ranges, if present.



7. To remove all variant expressions, click **View menu** and choose **Clear Expression**.

   The system re-evaluates the configuration and displays features that were hidden based on the previous selections.

8. Set the type of violation messages to be displayed by the system as follows:

   - Click **Set Info level violations to be fetched while applying validations** to switch between displaying and enforcing or hiding and dropping of all informational messages.
   - Click the **Set Warning level violations to be fetched while applying validations** to switch between displaying and enforcing or hiding and dropping of all warning messages.

   Error level violations are always displayed.

   The severity of the rule is honored when the system displays the available features in active mode. For example, if you set a minimum error severity of *warning*, then the features that violate the *information* severity rules are also available for selection in active mode along with the features that do not violate any other rules.

9. Select families, one group at a time, making modifications and resolving violations.

Click ▲ or ▼ to switch groups to select families, one group at a time.

If you deselect the **Allow Validation Rules to Expand** button, system selections based on validation rules does not occur after selection in the guided mode.

Teamcenter applies constraints to each group according to your previous selections. Each time you select a different group, it:

- Displays only valid available features. Features that are not available are hidden.
  - If no model is selected, the system shows all features that are available to any model in the configurator context.
  - Once a model is selected, the system only shows the features that are available to the selected model.
  - Similarly, if you select a feature prior to selecting any models, then the system only shows the models where that feature is available.

  > Note:
  >
  > Because configuration validation is automatically done whenever you navigate to a different group, you can make modifications to resolve the violations.

- Validates the current selections and identifies the validation result with appropriate symbols.
- Determines valid selections in the next group and applies constraints.

  If there are no valid selections for a family in the next group, that family is not included. If the family is required, it displays an error message.

  The system preselects features based on defined configurator rules. It also hides the features that are excluded as a result of constraints. If multiple features apply, the system does not suggest a selection. In this case, you can create a default configurator rule to define a preferred selection.

  Selection of features through inclusion rules during configuration rules out other features. Whereas selection of features from default rules does not rule out other features. You can only select features that are available and selectable, if no other rules create any conflicts.

- Applies system-assigned features and defaults to the families in the next group.

  You can change the selection from a system-assigned to a user-assigned feature. For example, if a **Music Player** feature is set to a system-assigned **DVD** feature, you can choose an **MP3** feature instead, as an upgrade.

The system allows you to validate configurations by specifying operators only for free-form families. You can only use = and != operators for text and free-form string features.

> Note:
>
> While computing a valid and complete configuration in the guided mode, the system selection of features within families in a group is only honored for that group. Switching to a new group makes the system selection ineffective for the previously selected groups. After switching groups during the guided mode, you can switch to the manual mode to view the system-selected features that impact the completeness of your configuration.

10. If you do not want to apply constraints, deselect **Apply constraints**. By default, this option is selected.

    - If selected, constraints are applied while applying the variant rule expression on the content.
    - If deselected, only user selections are applied.

11. If you do not want validation rules to participate during criteria expansion, deselect **Allow validation rules to expand**. By default, this option is selected.

    - If selected, allows validation rules to participate during criteria expansion.
    - If deselected, only violations, if present, are reported. The system selections based on validation rules does not occur in the expanded criteria.

    For more information, see **Example: Apply constraints and allow validation rules to expand**.

12. Click  to save the custom rule.

    If you started from a blank view, Teamcenter displays the **Save Product Configuration** dialog box; otherwise it saves the changes to the existing configuration.

    If you exit the **Variant Configuration** view without saving, you are prompted to save the changes.

13. To save a new configuration, in the **Save Product Configuration** dialog box, type a name and description, and click **OK**.

    Teamcenter creates a custom product configuration with the specified name and variant configuration. The system adds the name of the saved variant rule to the title bar.

14. To clear previously loaded custom configuration, click **View menu**  and choose **Clear Variant Rule**.

    The system removes the name of the saved configuration from the title bar and then resets user and system selections to the initial state.

15. (Optional) Click **View menu**  and choose **Clear Un-configured Feature** to remove unconfigured features.

The expression on the unconfigured feature in the view is removed. Also, the row holding the unconfigured feature is removed.

Configure using variant rules

Variant rules are created with the Product Configurator application by saving the configuration.

- In Product Configurator, click **Open Saved Variant Rules view** on the main toolbar.
  - In the **Saved Variant Rules** view, right-click the variant rule or variant criteria and choose **Open with→Variant Configuration**.

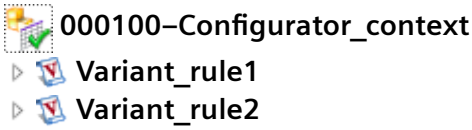
    > Note:
    >
    > If loading revised variant criteria, Teamcenter uses a default revision rule set in the **Variability Explorer** view to display the applicable revision of variant criteria in the **Saved Variant Rules** view.

    Teamcenter opens the **Variant Configuration** view and displays both the user and system-assigned selections along with the session information, as saved in the variant rule. The header in the view displays the name of the variant rule or the ID, the revision, and the name of the variant criteria.

    Teamcenter reconfigures the selected elements with the applied variant rule.

- In My Teamcenter, search for the configurator context containing the variant rules. You can also search to find variant rule of variant criteria.

  1. Expand the configurator context to display the saved variant rules and variant criteria, including all revisions, if any.

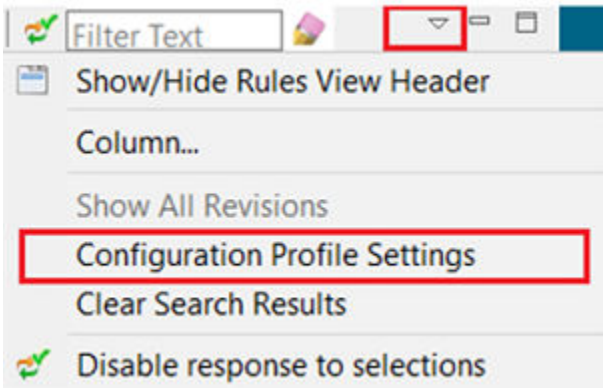
  2. Right-click the variant rule or variant criteria and choose **Open with→Variant Configuration**.

     Teamcenter reconfigures the selected elements with the applied variant rule.

     The loaded configuration contains both the user and system-assigned selections, as saved in the variant rule.

Set the session information for new variant rules and variant criteria

You can set the session information for new variant rules and variant criteria in the **Saved Variant Rules** view by clicking **View menu** and then selecting **Configuration Profile Settings**. It displays the **Configuration Profile Settings** for new **Variant Rules** dialog.



Button	Description
Validation mode	Select the validation mode for the new variant rules: **Order** or **Overlay**. In **Overlay** mode, rules greater than or equal to validation severity are evaluated. Rules for validation verdict and expand operations in overlay mode are same. In **Order** mode, during the expand operation, rules lower than validation severity are dropped if they conflict with rules of higher severity. The validation mode in the dialog defaults based on the value of the **Cfg0DefaultValidationMode** user preference. By default, it is set to **Order** mode.
Validation severity	Select the severity of violation for the new variant rules (error, warning and info). The error severity is selected by default.
Apply Constraints	Select if all constraints should be considered while applying the variant rule expression on the content. If deselected, only user selections are applied. By default, it is selected in the dialog.
Allow Validation Rules to Expand	Select if validation rules should participate during criteria expansion. If deselected, only violations are reported, if present. System selections based on validation rules do not occur in the expanded criteria. By default, it is selected in the dialog.

While creating a new variant rule or variant criteria, you can choose the profile settings from the dialog. The selected values are saved as session information on the newly created variant rule or variant criteria.

You cannot modify the profile setting while updating the variant rule or variant criteria in the **Saved Variant Rules** view. To update the profile settings for the variant rule or variant criteria, it should be opened in the **Variant Configuration** view.

Any changes made in the default settings of configuration profile settings dialog are persisted till the **Saved Variant Rules** view is open. When a new instance of the view is opened, the selections are reset to their default values.