Price rules

Marketing tools (8:10)
Play
Price Rules

Overview SaaS 3.5.0+

Price rules allow large reseller businesses better manager their price lists by configuring a set of rules to transform raw prices to end customer prices.

Typical use case is when reseller receives daily buying in price updates from suppliers which could contain thousands of products and it simply is impractical to review the end customer prices on daily basis manually. Instead resellers can setup rules such as: "HP Notebooks should be sold with 5% margin" or "All notebook accessories should be sold with 15% margin" which could be applied to buying in prices to automatically work out the correct end customer price.

Another typical case is when resellers receive RRP as a daily update from their supplier and end customer price is discounted from the RRP to give realistic customer prices. Example rule would be: "All cameras are sold with 10% discount from RRP"

Price rules management section allows to configure such rules and then transform the raw buying in prices into the end customer prices automatically by running **Price generator** job.

Price calculation

When price generator job executes it scans all non-auto-generated prices (prices which do not have auto-generated flag). Each price is evaluated by the **price rule engine** using set of price rules configured for the shop the price comes from (as each price belong to a specific shop).

Each rule has an **eligibility condition** which determines if a price is applicable for this rule and if it is the action of this rule is executed. The first applicable rule determines what happens to raw price in terms of action, which is why **ranking** of the price is very important as multiple rule could have condition which makes the price eligible but only the action of the first rule applicable is executed against the price.

The following actions are supported by price rules:

Action	Behaviour	Parameters used
Calculate	Calculates end customer price using formula: PRICE = RAW * (1 + MARGIN/100) + AMOUNT	Margin Percent - margin percentage (Can be negative for discount calculation) Margin Amount - fixed amount (Can be negative for discount calculation) Add tax - flag, when enabled tax is resolved an added to generated price (e.g. when raw prices are NET but shop works with GROSS prices by default) Rounding unit - minimal unit for "nice" prices rounding.

Request for price	Behaves similarly to Calculate but also adds the request for price flag onto the price. This flag prevents customer from seeing the price and instead a label is displayed that they need to contact the shop for price quote	Same as for Calculate action
Skip	Ignore the price, which is useful to skip generation of customer price (effectively saying we are not selling these products)	

B2B

In B2B configurations sub shops inherit all price rules from the master shop but also can define additional rules. Typically this is used to account for special conditions for sub shop, such as "Products from category X are not sold to this customer", or special prices such as "Customers in this sub shop receive a larger discount".

It is also possible to restrict sub shop to use only its own rules by setting **SHOP_B2B_STRICT_PRICE_RULES** attribute to **true** at sub shop level.

Tag, Reference and Policy

With **calculate** and **request for price** you have the option to set **tag, reference** and **policy** on the generated price. **Tags** are useful to track the origin of the generated price as you can see tags on the generated prices in the price lists view once they are generated, thus rule that caused the generation of this price can be easily identified. **Reference** is a special field on the price which is copied over to the cart items and later on to the placed order items (so it is like a sticky tag, which you will be able to see in order view). **Policy** is special field on prices that allows to limit access to these prices to only customers that have this policy set on their profiles (effectively exclusive right to special prices).

Eligibility condition

Eligibility condition is a logic statement that allows to evaluate an input price and determine if it is applicable for given rule. Each condition evaluates either to true or false. The condition syntax may seem somewhat overwhelming but in essence it is not much more complex than learning Excel formulas. However if this syntax is mastered marketing manager can create some very powerful and complex conditions to fine tune the pricing policies pitch perfect.

The eligibility condition editor provides helper functions for including a typical condition templates, looking up ID of categories and ID of brands which can be used as variable in the condition function.

Watch out for

Eligibility condition is a boolean expression (i.e. expression that evaluates either to true or false) which represents the qualifying criteria for price rule. Default rule expression engine is Groovy (a java library).

Since the expression written in Groovy here are some not so obvious things:

Comparing values: **A = B**	Equals operator is "==" (double equals) or .equals()	`// Use double equals sign or .equals()`
		`A == B`
		`A.equals(B)`

Complex statements	The **condition is a script** and can have multiline Groovy code as long as the last line evaluates to true or false	`// Check item SKU is one in the list of promo SKU codes`
		`def list = ['PROMOSKU001', 'PROMOSKU002', 'PROMOSKU003'];`
		`list.contains(shoppingCartItem.productSkuCode)`

Built in variables	There are some **predefined variables**t hat can be used in expression	`// customerTags is variable containing list of tags from Customer profile` `customerTags.contains('bigspender')`

Using variables and Groovy syntax any condition can be written to represent price rule with conditions resembling a natural English language sentences.

Variables

Variable	Type	Example	Description

PRICE	SkuPrice	`// Pricing policy check`
		`PRICE.pricingPolicy == 'COST_MAIN'`
		`// Pricing policy check for all policies starting with COST_`
		`// '?' character must be used since policy may not exist`
		`PRICE.pricingPolicy?.startsWith('COST_')`
		`// Zero prices`
		`PRICE.regularPrice == 0`
		`// Tag check`
		`PRICE.tag == 'special'`
			SkuPrice object being evaluated

SKU	String	`// match`
		`SKU == 'ABC'`
		`// partial match e.g. ABC-0001, ABC-0002`
		`SKU.startsWith('ABC')`
		`// partial match e.g. 00001-ABC, 00020-ABC, XYZ-ABC`
		`SKU.endsWith('-ABC')`
			SKU code of currently evaluated price

Functions

Function	Return type	Parameter 0	Parameter 1	Example
hasProductAttribute	boolean	String (SKU code)	String (Attribute code)	`hasProductAttribute(SKU, 'ONSALE')`
productAttributeValue	String	String (SKU code)	String (Attribute code)	`productAttributeValue(SKU, 'ONSALE') == 'Y'`

isSKUofBrand	boolean	String (SKU code)	String Array (Brand names)	`// Check for single brand`
				`isSKUofBrand(SKU, 'HP')`
				`// Check for multiple brands`
				`isSKUofBrand(SKU, 'HP', 'Lenovo')`

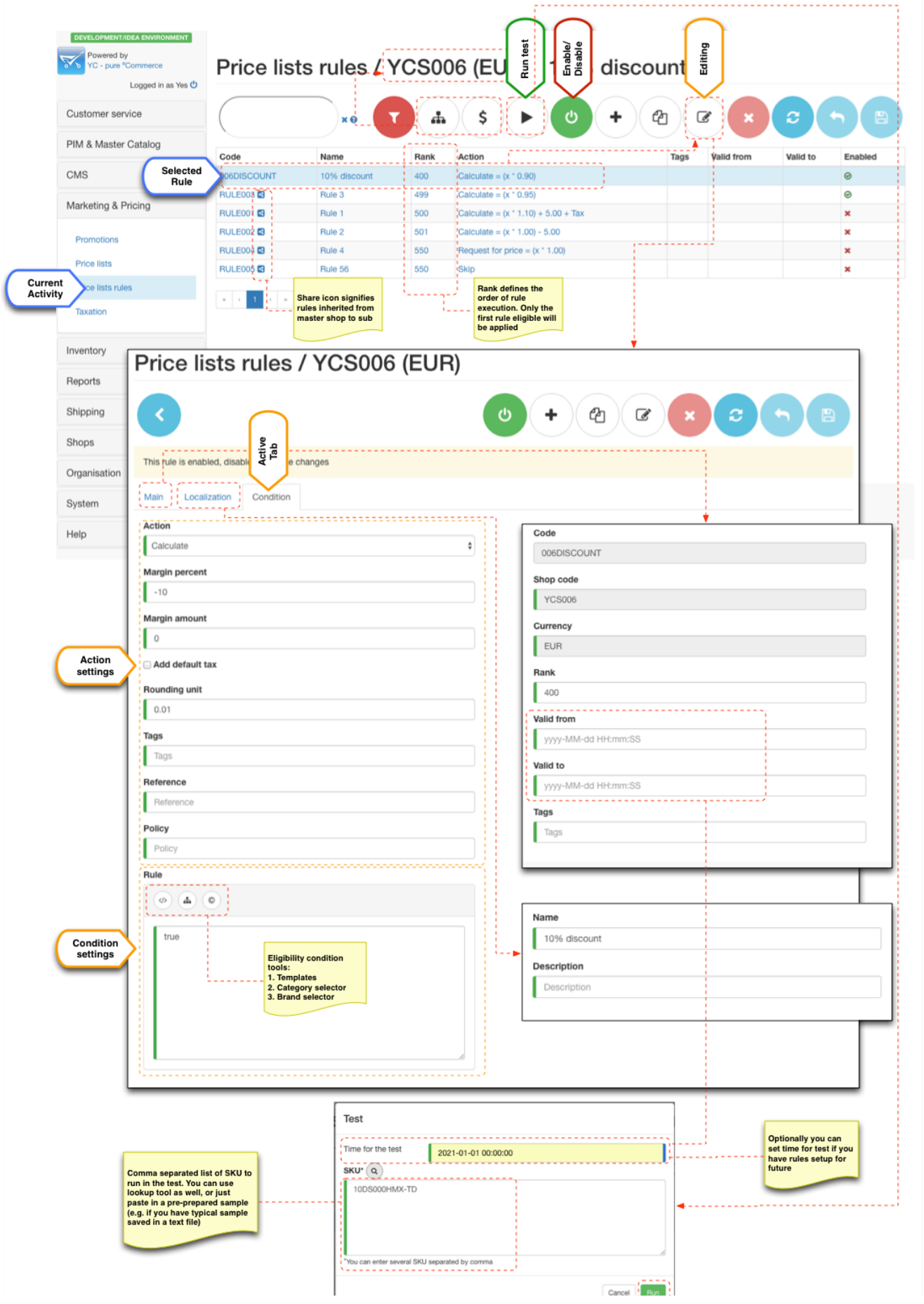
isSKUinCategory	boolean	String (SKU code)	String Array (Category code)	`// Check for single category`
				`isSKUinCategory(SKU, 'Notebooks')`
				`// Check for multiple categories`
				`isSKUinCategory(SKU, 'Notebooks', 'Accessor`

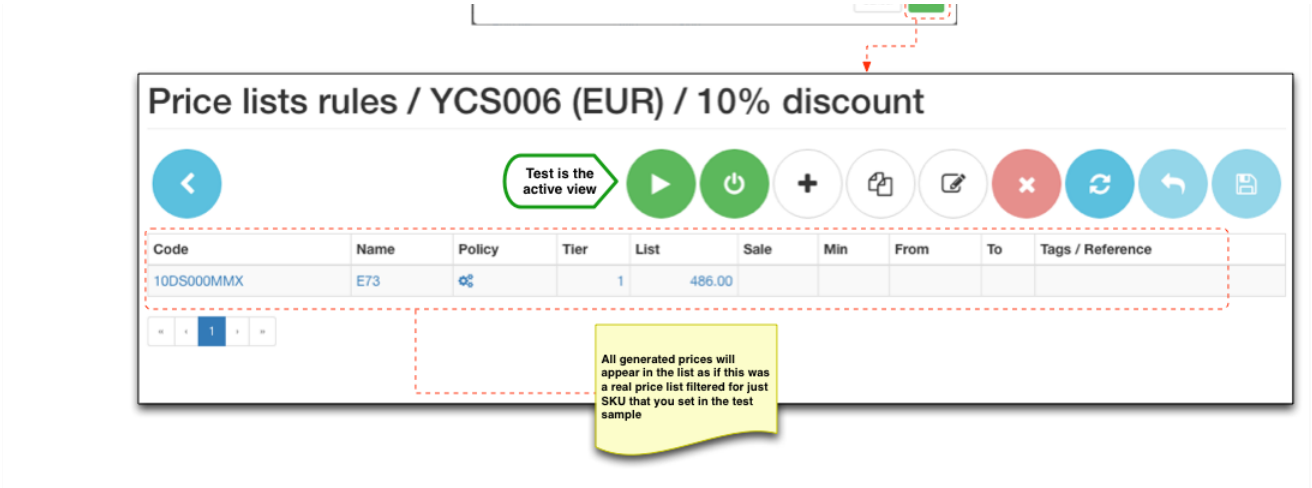
product	Product	String (SKU code)		`product(SKU).name == 'E73'`
productSku	ProductSKU	String (SKU code)		`productSku(SKU).name == 'E73'`
brand	Brand	String (SKU code)		`brand(SKU).name == 'HP'`

Price rules tester

Price rules management section includes price rules tester function which allows to run the configured rules on a set of SKU code and allow you to see the exact result of the calculations. You can also set the time variable to see how the rules behave in different time periods which is ideal for testing how the rules are evaluated say in seasonal sales and how calculations are performed with time sensitive rules and time sensitive raw (input) prices.

Price Rules Management


DEVELOPMENT/IDEA ENVIRONMENT
Powered by
YC - pure nCommerce
Logged in as Yes
Customer service
PIM & Master Catalog
CMS
Marketing & Pricing
Promotions
Price lists
Price lists rules
Taxation
Inventory
Reports
Shipping
Shops
Organisation
System
Help
Current Activity
Selected Rule
Price lists rules / YCS006 (EU
discount
Run test
Enable/ Disable
Editing
Code
Name
Rank
Action
Tags
Valid from
Valid to
Enabled
006DISCOUNT
10% discount
400
Calculate = (x * 0.90)
RULE003
Rule 3
499
Calculate = (x * 0.95)
RULE001
Rule 1
500
Calculate = (x * 1.10) + 5.00 + Tax
RULE002
Rule 2
501
Calculate = (x * 1.00) - 5.00
RULE004
Rule 4
550
Request for price = (x * 1.00)
RULE005
Rule 56
550
Skip
Share icon signifies rules inherited from master shop to sub
Rank defines the order of rule execution. Only the first rule eligible will be applied
Price lists rules / YCS006 (EUR)
Active Tab
This rule is enabled, disabl
e changes
Main
Localization
Condition
Action
Calculate
Margin percent
-10
Margin amount
0
Add default tax
Rounding unit
0.01
Tags
Tags
Reference
Reference
Policy
Policy
Rule
true
Action settings
Condition settings
Eligibility condition tools:
1. Templates
2. Category selector
3. Brand selector
Code
006DISCOUNT
Shop code
YCS006
Currency
EUR
Rank
400
Valid from
yyyy-MM-dd HH:mm:SS
Valid to
yyyy-MM-dd HH:mm:SS
Tags
Tags
Name
10% discount
Description
Description
Test
Time for the test
2021-01-01 00:00:00
SKU*
10DS000HMX-TD
You can enter several SKU separated by comma
Cancel
Run
Comma separated list of SKU to run in the test. You can use lookup tool as well, or just paste in a pre-prepared sample (e.g. if you have typical sample saved in a text file)
Optionally you can set time for test if you have rules setup for future



Working example

Business scenario

Given typical use cases described above suppose we have price import feeds available from our suppliers and we are able to receive **buying in** prices (BP) and **recommended retail prices** (RRP). Suppose that our BP are provided as NET (without tax) and RRP as GROSS (tax included) and our basic price view for web shop is GROSS prices.

Our marketing strategy for price would be to utilise both types of raw prices and devise a strategy based on the following rules:

- Notebooks (products in category 'notebooks') should be sold as 15% margin from BP
- Lenovo brand products should be sold as 5% discount from RRP
- We do not sell products from mobile category (say we receive products in PIM feed but we do not want to sell them in our shop)

Price import

Raw prices can be imported utilising the CSV import facility available as standard. Auto import listener job can be configured to pick up raw prices CSV import files at designated time.

However we do not want to expose these prices on the front end as both RRP and especially BP are not meant to be accessible by customers. Therefore we will use a special **policy** (this is an optional data field that exists on every price and if set will only be visible to customers that have this policy set on their profile).

Recommended usage for BP policy is 'COST_' + fulfilment centre code . For example if you FF code is MAIN, then the BR pricing policy field should be filled in with COST_MAIN . If you are following this convention then the BP prices will be automatically be resolved for the items that your customer purchases and will be visible in JAM's order view (provided you have access to this).

Similarly to BR, RRP policy could be set to 'RRP_' + fulfilment centre code .

Therefore the raw prices import file would look something like this (this is standard format for skuprices.xml import descriptor):

SKU code	Shop code	currency	quantity	list price	sale price	valid from	valid to	tag	pricing policy	ref
NB-0001	SHOPX	EUR	1	500					COST_MAIN	
NB-0001	SHOPX	EUR	1	750					RRP_MAIN	
NB-0002	SHOPX	EUR	1	520					COST_MAIN	
NB-0002	SHOPX	EUR	1	700					RRP_MAIN	
LE-0001	SHOPX	EUR	1	430					COST_MAIN	
LE-0001	SHOPX	EUR	1	580					RRP_MAIN	
MOB-0001	SHOPX	EUR	1	250					COST_MAIN	
MOB-0001	SHOPX	EUR	1	410					RRP_MAIN	

...										

> The import file can be single file, or can be broken into several files (e.g. BP and RRP separate) or can be any parts of the BP/RRP set or a mix. In terms of import this is not relevant as long as all columns are specified correctly.

> Above is the standard import for price file. It is possible to configure own format for the import descriptor including only columns you need (even on per shop basis). See import/export documentation for more details

Price rules

We setup the following price rules that reflect our business requirements:

Code	Rank	Eligibility condition	Action	Margi Perce
NOSALE	1	`(PRICE.pricingPolicy == 'COST_MAIN') && (isSKUinCategory(SKU, 'Mobile')`	Skip	
NB15MARGIN	2	`(PRICE.pricingPolicy == 'COST_MAIN') && (isSKUinCategory(SKU, 'Notebooks', 'PortablePC')`	Calculate	15
LE5DISCOUNT	3	`(PRICE.pricingPolicy == 'RRP_MAIN') && (isSKUofBrand(SKU, 'Lenovo')`	Calculate	-5

As you can see we rearranged the rules slightly and given NOSALE rule top rank, because this is the most fundamental rule and any price which matches this condition should be ignored as we do not sell these product and if no price is generated this product will not appear in the front end (all products must have a valid price unless they are set as SHOWROOM availability).

Next NB15MARGIN acts only on prices whose pricing policy is COST_MAIN and SKU is in notebook categories. If the price being evaluated satisfies this condition the calculation action is applied to generate a price with 15% margin and tax added to the result. Effectively we convert our BP which was NET into customer end price as GROSS (including tax). Say tax for this type of products is 20% then the calculation for NB-0001 would be (500 * (1 + 15/100)) * (1 + 20/100) = 690

> The tax applied would be default tax for given product. This particular example works in VAT type systems were tax for products is deterministic. For US/Canada taxation we recommend using NET prices

Last rule LE5DISCOUNT act only on prices whose pricing policy is RRP_MAIN and is of Lenovo brand. The calculation for LE-0001 would be (410 * (1 + (-5)/100)) = 389.50

Important to mention is that Lenovo notebooks would be calculated using NB15MARGIN rule as it has higher rank and will be applied first. So rules should be arranged by rank in such a way to produce correct prices. The question is with this scenario, what is the right rule to apply NB15MARGIN (because it is a notebook product) or LE5DISCOUNT (because it is of brand Lenovo). These types of questions can only be answered by marketing manager and thus it is up to them to arrange these rules in the correct ranking.