Formulas Cheat Sheet

Last Modified on 02/27/2019 2:20 pm EST

The cheat sheet has gotten a big upgrade! Click here for the new cheat sheet - now with more examples, lots more information about syntax and usage, and a more searchable format. Note that as of July 2019, this version of the cheat sheet won't get any new formula updates.

Skip to formula syntax & definitions, or use the links below to learn more about using formulas.

Want a downloadable version of the cheat sheet? Visit the cheat sheet catalog, and choose Export Catalog for an Excel spreadsheet.

Product Content Formula How-To & Examples

- Formulas Overview - What formulas are and why they're used.
- Anatomy of a Formula - Step by step details about how the pieces of a formula work. Includes a video walkthrough.
- Formula Building Basics - Learn how to use the formula editor and about the basic types of formulas and their syntax.
- Using Variables - Simplify and streamline your formulas with variables.
- Common Formula Use Cases - Example formulas and step-by-step help for building formulas.

Best Practices & Troubleshooting

- Best Practices & Guidelines
- Formula Troubleshooting

Other Transformation Formulas

- Image Transformations
- Video Transformations
- Renaming Digital Asset Files

Product Content Formulas

Use this reference to find specific formulas and syntax. Click a button to filter to specific formula types, sort by any column, and as you type in the search box, the results will automatically narrow to those that match your text.

Usage

Check out Anatomy of a Formula for full details, including a step-by-step video, on how the pieces of a formula work. Most formulas can be used in readiness reports, templated exports and computed properties, with the following exceptions:

- **Computed Properties** - `LOOKUP`, `TODAY`, and references to other computed properties are not supported.

Copying & Using Formulas from the Cheat Sheet

Formula & Syntax Column

When copying a formula from the table below, replace what's inside , including those brackets. So for example, for `VALUE("")` , if your property ID is *Brand,* you replace including the brackets, and the formula would be `VALUE("Brand")` .

Examples & Output Column

Refer to the output below the formulas for what to replace. The pieces in italics are what you replace with your own values. In this example:

`CONCATENATE(VALUE("Brand"), ": ", VALUE("Product Name"))`
Where *Brand* is Acme and *Product Name* is Anvil, output is Acme: Anvil

Replace *Brand* and *Product Name* with your property IDs.

The cheat sheet is also available in a Salsify catalog, with an Excel download. Click here to visit the cheat sheet catalog.

Show All Categories | Pulling Out Values | Conditionals | Combining & Transforming Values | Working With Numbers | Category Hierarchy Values | Digital Asset URLs & Metadata | Arrays & Advanced Formulas

Formula Type	Formula & Syntax	Definition	Example & Output
Working with Numbers	**ADD** `ADD("<propertyID>", "<propertyID>"...)`	Arithmetic function applied to numeric property values or numbers to obtain calculated values.	`ADD(VALUE("MSRP"),VALUE("Surcharge")` Where *MSRP* is 19.99 and *Surcharge* is 2.00, output is 21.99.
Conditionals	**AND** `IF(AND("<propertyID1>" *or* "<formula1>", "<propertyID2>" *or* "<formula2>")`	Allows testing two values in relation to each other in a formula. Both values must be true for the result to be true.	`IF(AND(EQUAL(VALUE("Country of Origin"), "United States"), EQUAL(VALUE("Country of Assembly"), "United States")), "Made in USA", "Imported")` Where both *Country of Origin* and *Country of Assembly* are United States, output is Made in the USA. If both are not United States, output is Imported.
DIgital Asset URLs & Metadata	**ASSET** `ASSET("<your digital asset_metadata_property id>", <optional_index>)`	Returns the value of the specified metadata attribute for the Nth (optional) digital asset in the product. If index isn't specified, formula returns value for the first asset.	`ASSET("Manufacturer")` Where Manufacturer's property value is Wildflower Imports, output is Wildflower Imports
DIgital Asset URLs & Metadata	**ASSET_VALUE** `ASSET_VALUE("<propertyID>","<metadata attribute>","<N>")`	Returns the value of the attribute for the Nth digital asset in the property. System metadata properties must be prefixed with salsify:. Click here for the list of Salsify system metadata properties.	`ASSET_VALUE("Hero Image", "salsify:Height", 1)` Where the first digital asset in *Hero Image* has a height of 500, output is 500.
Arrays & Advanced Formulas	**AT** `AT(["<propertyID1>", "<propertyID2>", "<propertyID3>"], 3)`	Allows you to return the Nth value of an array.	`AT(["Red", "Blue", "White"], 3)` Output is White.
Pulling Out Values	**CHANNEL_STATUS_FROM** `CHANNEL_STATUS_FROM(<channel_id>)`	Returns channel status string value for the most publish, if it exists.Returns empty/null value if no status is available from the channel,or product has never been published through the	`CHANNEL_STATUS_FROM(11234)` Where current channel status is "Published", output is "Published".

Formula Type	Formula & Syntax	Definition	Example & Output
		channel. To find channel ID, navigate to the channel and the ID is in URL path after /channels/.	
		Function is not currently available for in-app computed properties.	
Pulling Out Values	**CLEAN_TEXT** `CLEAN_TEXT(VALUE("<propertyID>"))`	Removes any special and/or hidden characters, including trademarks and/or copyright symbols.	`CLEAN_TEXT(VALUE("Brand"))` Where *Brand* is Acme™, output is Acme.
Combining & Transforming Values	**COALESCE** `COALESCE("<propertyID1>", "<propertyID2>", ...)`	Outputs the first value given that is not empty.	`COALESCE(VALUE("Amazon Item Name"),VALUE("Retail Item Name"),VALUE("ERP Item Name"))` Output is the *Amazon Item Name* value. If it's empty, output is *Retail Item Name* value. If empty, *ERP Item Name* value is the output.
Arrays & Advanced Formulas	**COMPACT** `COMPACT("<formula>")`	Removes blank values that would be generated by empty properties.	`COMPACT(CONCAT_ARRAYS(VALUES("Feature 1"), VALUES("Feature 2"), VALUES("Feature 3")))` Where *Feature 1* is Blue, *Feature 2* is blank, *Feature 3* is 2 pack, returns Blue and 2 pack on separate lines and doesn't leave a blank line for Feature 2.
Arrays & Advanced Formulas	**COMPOUND** `COMPOUND("<label1>", "<propertyID1>","<label2>", "<propertyID2>"...)`	Specify pairs of information, a label and the value that the information is stored in. Output is label:property value	`COMPOUND( "Color Family",VALUE("Color"), "Fabric",VALUE("Material"))` Where *Color* value is Blue and *Material* value is Cotton, output is: Color Family:Blue Fabric:Cotton
Combining & Transforming Values	**CONCATENATE** `CONCATENATE("<propertyID1>", "<propertyID2>", ...)`	Combines together multiple values. You can concatenate as many values as you want, and you can combine literal text values with other Salsify formulas.	`CONCATENATE(VALUE("Brand"), ": ", VALUE("Product Name"))` Where *Brand* is Acme and *Product Name* is Anvil, output is Acme: Anvil
Arrays & Advanced Formulas	**CONCAT_ARRAYS** `CONCAT_ARRAYS(VALUES("<propertyID1>","<propertyID2>"...)`	Use with VALUES to return multiple values from each referenced property	`CONCAT_ARRAYS(VALUES("Features"), VALUES("Benefits"))` Where *Feature 1* is shiny, *Feature 2* is bright, *Benefit 1* pretty, and *Benefit* 2 is kind, output is: shiny bright pretty kind.
Conditionals	**CONTAINS** `IF(CONTAINS("<propertyID to test>", "<string to check>"), "<true_result>", "<false_result>")`	Tests a value for the string specified in the argument. If the string exists in the value, result is true; if not, false.	`IF(CONTAINS(VALUE("Brand"),"Salsify"),"Cool","Bummer")` If *Brand* contains Salsify, output is Cool. If not, Bummer.
Pulling Out Values	**CREATED_AT** `CREATED_AT()`	Returns the date product record was created, in UTC in the format yyyy-mm-dd.	`CREATED_AT()` Where record was created April 24, 2018, output is 2018-04-24.
Pulling Out Values	**DATE_LAST_PUBLISHED_TO** `DATE_LAST_PUBLISHED_TO(<channel_id>)`	Returns date of most recent publish through the channel indicated via the publish button in the channel, through	`DATE_LAST_PUBLISHED_TO(11234)`

Formula Type	Formula & Syntax	Definition	Example & Output
		ephemeral publish or marked as publish event via the readiness report. If product has notbeen published, result is empty/null.To find channel ID, navigate to the channel and the ID is in URL path after /channels/. Function is not currently available for in-app computed properties.	Where the most recent product publish date was 5/24/17, output is 2017-05-24.
Arrays & Advanced Formulas	**DIFFERENCE** `DIFFERENCE(["<array1>"], ["<array2>"])`	Allows you to subtract one array from another to return only the remaining values.	`DIFFERENCE(["White","Blue","Red"],["Blue","Red"])` Output is White.
Working with Numbers	**DIVIDE** `DIVIDE("<propertyID1>", "<propertyID2>")`	Arithmetic function applied to numeric property values or numbers to obtain calculated values. Accepts 2 values.	`DIVIDE(VALUE("Cost"),VALUE("Retail")` Where *Cost* is 10 and *Retail* is 25, output is .4
Arrays & Advanced Formulas	**EACH** `EACH(<"array">,(<name for single item in array>,<*index*>) => <"action you'd like to take">)`	Access each value in an array. To use, define the array, name for each item within the array, and the action to take on each. Typically used in combination with other formulas. Optionally, include index to add a number corresponding to each item's position in the array.	`EACH(VALUES("Bullet Points"), (feature,index) => CONCATENATE("Bullet",index,": ",feature))` Where Bullet Points contains "Red" and "Blue", output is: Bullet 1: Red Bullet 2: Blue
Conditionals	**EQUAL** `EQUAL("<propertyID1>", "<propertyID2 *or* specific_numeric_value>")`	Allow you to compare two values, and can be used with conditional formulas to evaluate true/false state. Can be used in combination with IF. Returns true if value is equal to the compared value.	`IF(EQUAL(VALUE("Assembly Required?"), "Yes"), "Y", "N")` If *Assembly Required* is Yes, output is Y. If it is blank or other than Yes, output is N.
Pulling Out Values	**FIND** `FIND("<search string>", "<search_location>")`	Finds a string of characters inside a specified search location, in most cases a property value or variable. Combine with other formulas to perform actions on found string.	`IF(FIND("Soft Line",VALUE("Category")),"Apparel","Hard Goods")` If *Category* contains Soft Line, output is Apparel, otherwise output is Hard Goods.
Combining & Transforming Values	**FORMAT_DATETIME** `FORMAT_DATETIME(<"current_date value">, <"current date parameters">, <"optional transformed_date_parameters">)`	Transforms a date from one format to another. Where transformed date is not included, default is Salsify format (YYYY-MM-DD). Include the current date value, either fixed or stored, and define the format where `%m` = month, `%d` = day, `%Y` = year. Uses strftime formatting. Include delimiters in the date parameters to define the separate the date parts.	Where the date you want to convert is stored in a Salsify date-formatted property called Date Property, formula would be `FORMAT_DATETIME(VALUE("Date Property"),"%Y-%m-%d", "%m/%d/%Y")` If stored value were 2019-02-26, output would be 02/26/2019.
Working with Numbers	**GE** `GE("<propertyID1>", "<propertyID2 *or* specific_numeric_value>")`	Greater than or Equal to. Allow you to compare two numbers/numeric values, and can be used with conditional formulas to evaluate true/false state. Can be used in combination with IF. Returns true if value is greater than or equal to the compared value.	`IF(GE(VALUE("Width"),"12"), "Large","Small")` If *Width* is 12 or greater, output is Large. If less than 12, output is Small.
Working with Numbers	**GT** `GT("<propertyID1>", "<propertyID2 *or* specific value>")`	Greater than. Allow you to compare two numbers/numeric values, and can be used with conditional formulas to evaluate true/false state. Can be used in combination with IF. Returns	`IF(GT(VALUE("Width","6"),"Large","Small")` If the value of *Width* is greater than 6, output is Large. If 6 or less,

Formula Type	Formula & Syntax	Definition	Example & Output
Conditionals	**IF** `IF("<test>", "<output if true>", "<optional_false_output>")`	Outputs either one value or another, depending on whether test is true or false. true and false outputs can be either literal values or Salsify formulas. This is similar to the IF formula in Excel.Tests either check if a product has a value for a particular property or check whether the value for a product's property is equal to a specific value.	`IF(VALUE("Brand"),"Acme")` If Brand contains a value, output "Acme", if not, output blank.
Arrays & Advanced Formulas	**IN** `IN("<searched text>", "<propertyID *or* array>")`	Returns T/F for whether searched text exists in the properties/array searched.	`IN("White","Red", "White", "Blue", "Green", "Yellow")` Output is true.
Working with Numbers	**ISNUMBER** `ISNUMBER("<propertyID>")`	Outputs true if value is a number, otherwise false. Typically used in combination with conditionals like IF.	`ISNUMBER("Hello")` Output is "False".
Working with Numbers	**IS_VALID_GTIN** `IS_VALID_GTIN(<"property *or* string">, <"gtin14, gtin13, gtin12, gtin8, or upc ">)`	Checks for a valid UPC (correct number of digits and valid check digit) or GTIN and returns a true/false value.	`IS_VALID_GTIN("Each GTIN","gtin12")` Where Each GTIN is valid, output is true. If Each GTIN is invalid, output is false.
Arrays & Advanced Formulas	**JOIN** `JOIN("<propertyID>", "<delimiter>")`	Outputs all values for a property with a specified delimiter between the values. Use with VALUES rather than VALUE.	`JOIN(VALUES("Bullets"), "\n")` Where *Bullets* values are red and blue, output is: red blue
DIgital Asset URLs & Metadata	**JOIN_ASSET_VALUES** `JOIN_ASSET_VALUES("<property name>", "<metadata property>", "<delimiter>")`	Joins metadata together for all of the digital assets associated with a property. Cannot be used for in-app computed properties.	`JOIN_ASSET_VALUES("Lifestyle Images", "salsify:Filename", ", ")` Where *Filename* values are DSC01.jpg and DSC02.jpg, output is DSC01.jpg,DSC02.jpg.
Category Hierarchy Values	**JOIN_PATH_IDS** `JOIN_PATH_IDS("<property ID>", "<optional delimiter>", "<optional_numeric_index>")`	Joins all levels of the hierarchy's IDs for the property together using delimeter specified, and for the property value specified by numeric index. Default delimiter is / and default index is 1. If your hierarchy properties include different value IDs and names, you can use the PATH_NAME function to access names. Also accepts an array for the index to pull back multiple values. Cannot be used for in-app computed properties.	`JOIN_PATH_IDS("Category", " | ", 2)` Where *Category* values are Bags > Laptop Bags > Laptop Backpacks Luggage > Wheeled Luggage > Wheeled Carry-on Luggage Output is Luggage | Wheeled Luggage | Wheeled Carry-on Luggage
Category Hierarchy Values	**JOIN_PATH_NAMES** `JOIN_PATH_NAMES("<property name>", "<optional delimiter>", "<optional_numeric_index>")`	Joins all levels of the hierarchy's names for the property together using delimiter specified, and for the property value specified by numeric index. Default	`JOIN_PATH_NAMES("Category", " | ", 2)` Where *Category* values are

Formula Type	Formula & Syntax	Definition	Example & Output
		delimiter is / and default index is 1. For hierarchy properties include different value IDs and names, you can use the PATH_ID function to access IDs. Also accepts an array for the index to pull back multiple values. Cannot be used for in-app computed properties.	Bags > Laptop Bags > Laptop Backpacks Luggage > Wheeled Luggage > Wheeled Carry-on Luggage Output is Luggage | Wheeled Luggage | Wheeled Carry-on Luggage
Pulling Out Values	**JOIN_PROPERTIES_FROM_GROUP** `JOIN_PROPERTIES_FROM_GROUP(<your property group&rt;', '<property/value separator&rt;', '<pair_separator&rt;')`	Returns all properties as a series of sets of values (property ID, value) with specified separators.	`JOIN_PROPERTIES_FROM_GROUP('General', ' - ', ')`The General property group contains two properties: Record ID which holds a value 123, and Record Name which holds a value Jetsetter Carry On. Output is Record ID - 123, Record Name - Jetsetter Carry On.
Combining & Transforming Values	**JOIN_RELATIONS** `JOIN_RELATIONS("<relation_label>", "<delimter>")`	Join all product identifiers of targets for a given relation label. Inputs are the relation label in Salsify and the separator you want to return.	`JOIN_RELATIONS("Also Bought", " ")` Returns all products listed under the relation label. Where the relation name is Also Bought, and the product IDs are 11111, 343434, and 454545, output is 1111 343434 454545. Click here for an example.
Pulling Out Values	**JOIN_VALUES** `JOIN_VALUES("<property name>", "<delimiter>")`	For properties with multiple values, combines all of the values together using the given delimiter.	`JOIN_VALUES("Feature Bullets", ", ")` Where *Feature Bullets* contains Big, Shiny and Economical, output is Big, Shiny, Economical.
Working with Numbers	**LE** `LE("<propertyID1>", "<propertyID2 *or* specific_numeric_value>")`	Less than or equal to. Allows you to compare two numbers/numeric values, and can be used with conditional formulas to evaluate true/false state. Can be used in combination with IF. Returns true if value is less than or equal to the compared value.	`LE(VALUE("Width"),"12")` If *Width* value is 12 or less, output is True, otherwise output is False.

Conditionals	**LENGTH**
	`LENGTH("<propertyID>")`
		Returns the number of characters for a fixed value, or specified property value when used with VALUE. In arrays (data sets enclosed in brackets, or through the use of VALUES), counts the number of values in the array.	Fixed value:
			`LENGTH("Apple")`
			Output is 5.
			
			Used with VALUE:
			`LENGTH(VALUE("Brand")`
			Where *Brand* value is Salsify, output is 7.
			
			Array example:
			`LENGTH([Apple,Banana,Carrot,Donut])`
			Output is 4.
			
			Array with VALUES:
			`LENGTH(VALUES("Tags")`
			Where *Tags* are apple and orange, output is 2.

Arrays & Advanced Formulas	**LET...IN**
	`let "<variable_name>"="<value or formula>" in`
		Use to define variables to be used in subsequent formulas. Variables can contain fixed values, or can perfom functions. Click here for more information on variables.	`let x=" " in`
			`let y=VALUE("Brand Name") in`
			
			`CONCATENATE(y,x)`

Formula Type	Formula & Syntax	Definition	Example & Output
		Note: when using in templated exports, don't include the typical `SALSIFY_` prefix for `IN`	Where *Brand Name* is Wildflower, output is Wildflower™
Combining & Transforming Values	**LOOKUP** `LOOKUP("<propertyID>", "<url or asset ID>", "<column header for value to return>", "<optional_default_value>")`	Looks up the value for a specified property, references a table stored in an FTP location or a Salsify asset IDand returns the corresponding value from the table. Table contains all the allowed values for the property in column A.	`LOOKUP(VALUE("Category"),"https://salsify.exavault.com/p/Enablement/Wildflower-categories.xlsx","SuperWidgets Category","General")` Where Category contains Food and in the xlsx file, the SuperWidgets column contains Grocery, output is Grocery.
Combining & Transforming Values	**LOOKUP_FIRST** `LOOKUP_FIRST("<propertyID>", "<url or asset ID>", "<2nd column_header>", "<default_value>")`	Differs from Lookup in that if there are multiple lookup keys found it will return the match for the first value found in the lookup table rather than error.	`LOOKUP_FIRST(VALUE("Category"),"http://salsify.exavault.com/p/Enablement/Wildflower-categories.xlsx","SuperWidgets Category","General")` Where Category contains Food on two rows, output is Grocery where the corresponding value in the lookup table at the URL indicated contains Grocery in the Superwidgets Category column the first time Food is encountered.
Combining & Transforming Values	**LOWER** `LOWER("<propertyID>")`	Converts the given value into all lowercase.	`LOWER(VALUE("Brand")` Where value for *Brand* is ACME, output is acme.
Combining & Transforming Values	**LPAD** `LPAD("<propertyID>", "<character_to_pad_with>", <N>)`	Outputs a value that is N characters long by adding at the front.	`LPAD(VALUE("UPC"), "0", "14")` Where *UPC* is 987654321, output is 00000987654321.
Working with Numbers	**LT** `LT("<propertyID1>", "<propertyID2 *or* specific_numeric_value>")`	Less than. Allows you to compare two numbers/numeric values, and can be used with conditional formulas to evaluate true/false state. Can be used in combination with IF. Returns true if value is less than the compared value.	`LT(VALUE("Width"),"6")` Where *Width* contains 6, output is False.
Combining & Transforming Values	**LTRIM** `LEFT("<propertyID>", <N>) or LTRIM("<propertyID>", <N>)`	Retains number of characters specified, starting from the left. Trims any additional characters at the right end of the string.	`LTRIM(VALUE("UPC"), 11)` Where UPC is 9876543210000, output is 98765432100
Pulling Out Values	**MATCHES** `MATCHES("<text to search>", "<search string or regex>")`	Finds a string of characters inside a specified search location, in most cases a property value or variable. Search string accepts REGEX. Combine with other formulas to perform actions on found string.	`MATCHES("I have $5.98 and you have $2.54", "\\$\\d+\\.\\d{2}")` Returns $5.98 $2.54
Working with Numbers	**MAX** `MAX("<propertyID1>", "<propertyID2>"...)`	Returns the largest numeric values for selected properties.	`MAX(VALUE("Length"),VALUE("Width),VALUE("Height"))` Where *Length* is 10, *Width* is 5 and *Height* is 2, output is 10.
Working with Numbers	**MEDIAN** `MEDIAN("<propertyID1>", "<propertyID2>"...)`	Returns the numeric value closest to the average for selected properties.	`MEDIAN(VALUE("Length"),VALUE("Width"),VALUE("Height"))` Where *Length* is 10, *Width* is 5 and *Height* is 2, output is 5.
Combining & Transforming Values	**MID** `MID("<propertyID>", ,`	Similar to MID in Excel. Outputs a substring of a value starting at	`PROPER(MID(VALUE("Feature Bullet"),9,23))`

Formula Type	Formula & Syntax	Definition	Example & Output
	<string_length>)	the Nth character/start index, and continuing for length of characters specified.	Where the value for *Brand* is Features padded shoulder straps and zippered pouch, output is Padded Shoulder Straps.
Working with Numbers	**MIN** MIN("<propertyID1>", "<propertyID2>"...)	Returns the smallest numeric values for selected properties.	MIN(VALUE("Length"),VALUE("Width),VALUE("Height")) Where *Length* is 10, *Width* is 5 and *Height* is 2, output is 2.
Working with Numbers	**MOD** MOD("<propertyID *or* specific_numeric_value>", "<divisor_N>")	Returns the remainder when the value is divided by the divisor.	MOD("158","10") Where value is 158, and divisor is 10, output is 8.0.
Pulling Out Values	**MODIFIED_SINCE_LAST_PUBLISHED_TO** MODIFIED_SINCE_LAST_PUBLISHED_TO(<channel_id>)	Returns true/false for whether any property value for the product has been modified since the most recent publish through the channel indicated. To find channel ID, navigate to the channel and the ID is in URL path after /channels/. Function is not currently available for in-app computed properties.	MODIFIED_SINCE_LAST_PUBLISHED_TO(11234) Where the product has been updated since most recently publishedthrough 11234, output is true.
Working with Numbers	**MROUNDUP** MROUNDUP("<propertyID>", <multiple_N>)	Returns the value rounded up by a given multiple.	MROUNDUP(VALUE("Price"),5) Where *Price* is 11.99, output is 15.0.
Working with Numbers	**MULTIPLY** MULTIPLY("<propertyID1>", "<propertyID2>"...)	Arithmetic function applied to numeric property values or numbers to obtain calculated values.	MULTIPLY(VALUE("Retail Price"),VALUE("Margin") Where *Retail Price* is 19.99 and *Margin* is .8, output is 15.991999999999999.
Conditionals	**NOT** NOT("<propertyID1, N1, string1 *or* formula1>", "<propertyID2, N2, string2 *or* formula2>")	Tests the relation between two numeric or string property values, formulas, or specific numbers or strings and returns a true or false. Typically used in combination with other conditional or numeric formulas.	NOT(EQUAL(12,6)) Because 12 and 6 are not equal, output is True.
Pulling Out Values	**NOW** NOW("<optional_format>")	Outputs the current time in iso8601 format by default (yyyy-mm-ddThh:mm:ss+timezone offset from UTC). Use strftime directives to modify format. Where current time is 10:00 AM on August 8, NOW() returns 2018-08-14T15:00:00+00:00 Optionally, use accepted time zone names.	NOW("%D") Where current time is 10:00 AM on August 8, output is 08/14/18
Conditionals	**OR** OR("<propertyID1, N1, string1 *or* formula1>", "<propertyID2, N2, string2 *or* formula2>")	Allows testing two values in relation to each other in a formula. If either statment is true, the output is true.	OR(EQUAL(12,6),EQUAL(6,6)) Because 6 is equal to 6, output is True.
Pulling Out Values	**PARENT_ID** PARENT_ID()	If product has a parent, outputs the Parent product's ID.	PARENT_ID() If the product's *Parent ID* is 1111, returns 1111.
Category Hierarchy Values	**PATH_ID** PATH_ID("<propertyID>",numeric_depth, *optional_numeric_index*)	For enumerated values with hierarchies, function returns the property value ID in the hierarchy level you specify. Use the optional numeric index in cases where you have multiple values stored and want to return other than the first property value. If index is not	PATH_ID("Category",3,2) Where Category contains two hierarchical value IDs Housewares > All Luggage > Soft-side Luggage Wheeled Luggage > Checked Luggage > Hard-side Luggage

Formula Type	Formula & Syntax	Definition	Example & Output
		included, function will pull from the first property value stored. Not available for use with in-app computed properties.	Output is Hard-side Luggage.
Category Hierarchy Values	**PATH_NAME** `PATH_NAME("<propertyID>",numeric_depth, optional_numeric_index)`	For enumerated values with hierarchies, function returns the property value name in the hierarchy level you specify. Use the optional numeric index in cases where you have multiple values stored and want to return other than the first property value. If index is not included, function will pull from the first property value stored. Not available for use with in-app computed properties.	`PATH_NAME("Category",2, 1)` Where Category contains the value names Housewares > All Luggage > Soft-side Luggage Wheeled Luggage > Checked Luggage > Hard-side Luggage Output is Wheeled Luggage.
Combining & Transforming Values	**PROPER** `PROPER("<propertyID>")`	Capitalizes the first letter of each word in the value.	`PROPER(VALUE("Brand")` Where value for *Brand* is ACME INC., output is Acme Inc.
Pulling Out Values	**PROPERTY_FROM_GROUP** `PROPERTY FROM GROUP('<your_property_group_name>', <index>)`	Returns the property and property value at the specified index of a property group with a custom separator applied if provided.	`PROPERTY_FROM_GROUP ('General', '- ', 2)`The General property group contains two properties: Record ID which holds a value 123, and Record Name which holds a value Jetsetter Carry On. Output would be Record Name - Jetsetter Carry On.
Pulling Out Values	**PROPERTY_ID_FROM_GROUP** `PROPERTY ID FROM GROUP('<your_property_group_name>', <index>)`	Returns the property id at the specified index of the property group.	`PROPERTY_ID_FROM_GROUP('General', 2)`The General property group contains two properties: Record ID which holds a value 123, and Record Name which holds a value Jetsetter Carry On. Output would be Record Name.
Arrays & Advanced Formulas	**PROPERTY_IDS_WITH_VALUE_FROM_GROUP** `property_ids_with_value_from_group("<property_group_id>")`	By default, returns all the labels of a Property Group as separate values. Combine with formulas like EACH, CONCAT and variables to return sets of labels and values. Function is not currently available for use with in-app computed properties.	`let rich_content = compact( each( property_ids_with_value_from_group("Rich Content"), external_id => if(value(external_id), concat( external_id,': ',value(external_id) ), null) ) ) in PROPER(rich_content)` Where the property group ID is Rich Content, property IDs are Material and Fabric Care, property values are 100% Cotton, Cold water machine washable, output is: Material: 100% Cotton Fabric Care: Cold Water Machine Washable

Formula Type	Formula Syntax	Definition	Example & Output
Pulling Out Values	**PROPERTY_NAME_FROM_GROUP** `PROPERTY_VALUE_FROM_GROUP('<your_property_group_name>', 2)`	Returns the property name at the specified index of the property group.	`PROPERTY_VALUE_FROM_GROUP('General', 2)`The General property group contains two properties: Record ID which holds a value 123, and Record Name which holds a value Jetsetter Carry On. Output would be Record Name.
Pulling Out Values	**PROPERTY_VALUE_FROM_GROUP** `PROPERTY_VALUE_FROM_GROUP('<your property_group_name>',<index>)`	Returns the property value at the specified index of the property group. Typically used where packaging hierarchies are stored.	`PROPERTY_VALUE_FROM_GROUP('General', 2)`The General property group contains two properties: Record ID which holds a value 123, and Record Name which holds a value Jetsetter Carry On. Output would be Jetsetter Carry On.
Pulling Out Values	**PUBLISHED_TO** `PUBLISHED_TO(<channel_id>)`	Returns true/false for whether the product has been published through the channel indicated via the publish button in the channel, through ephemeral publish or marked as a publish event via the readiness report. To find channel ID, navigate to the channel and the ID is in URL path after /channels/. Function is not currently available for in-app computed properties.	`PUBLISHED_TO(11234)` Where the product has been published through 11234, output is true.
Arrays & Advanced Formulas	**REDUCE** `REDUCE([array],formula, [optional initial value])`	`REDUCE` allows you to combine the values in an array one by one. To use it, you provide the array and a formula that will be run for each value in the array along with the previous result for that formula. For example, `REDUCE([1, 2, 3, 4], (sum_so_far, number) => ADD(sum_so_far, number))` takes each number in the array one by one, and adds it to the result of summing up the previous numbers in the array. It's the same as writing `ADD(ADD(ADD(1, 2), 3), 4)`, only it works no matter how many numbers are in the array. You can optionally give `REDUCE` an extra "starting value" to use, e.g. `REDUCE([1, 2, 3, 4], (sum_so_far, number) => ADD(sum_so_far, number),`	`REDUCE([1, 2, 3], (sum, number) => ADD(sum, number))` Where the array contains 1, 2 and 3, the output is 6.

Formula Type	Formula & Syntax	Definition	Example & Output
		`10)`, which is the same as `ADD(ADD(ADD(ADD(10, 1), 2), 3), 4)`.	
Pulling Out Values	**REGEX_MATCHES** `REGEX_MATCHES("<text to search>", "<search string or regex>")`	Finds a string of characters inside a specified search location, in most cases a property value or variable. Search string accepts REGEX. Combine with other formulas to perform actions on found string.	`REGEX_MATCHES("I have $5.98 and you have $2.54", "\\$\\d+\\.\\d{2}")` Returns $5.98 $2.54
Pulling Out Values	**RELATION** `RELATION("<relation_label>", <*optional*_index_N>)`	Returns product ID of target for a given relation label.	`RELATION("Other Colors",1)` Output is the product ID for the first related product listed in *Other Colors*,
Combining & Transforming Values	**REPLACE** `REPLACE("<propertyID>", "<find_text>", "<replace_text>")`	For our users familiar with regular expressions, the REPLACE function is just like SUBSTITUTE except it uses a regular expression.	`REPLACE(VALUE("Product Name"), "\n"," ")` Where *Product Name* value contains a line break, replace with a space.
Working with Numbers	**ROUND** `ROUND(VALUE("<propertyID>"), <N_decimal_places>)`	Rounds numeric values to the specified number of decimal places.	`ROUND(VALUE("Cost"), 2)` Where *Cost* is 19.992222, output is 19.99
Combining & Transforming Values	**RPAD** `RPAD("<propertyID>", <N_padding>, <N_length>)`	Outputs a value that is N characters long by adding at the end.	`RPAD(VALUE("UPC"), "0", "14")` Where *UPC* is 987654321, output is 98765432100000.
Combining & Transforming Values	**RTRIM** `RIGHT("<propertyID>", <N>)` or `RTRIM("<propertyID>", <N>)`	Outputs the last N characters of value.	`RTRIM(VALUE("UPC"), 11)` Where *UPC* is 9876543210000, output would be 76543210000
Combining & Transforming Values	**SENTENCE** `SENTENCE("<propertyID>")`	Capitalizes the first letter of the first word of a given value.	`SENTENCE(VALUE("Warranty")` Where value for *Warranty* is ten year limited warranty, output is Ten year limited warranty.
DIgital Asset URLs & Metadata	**SERIALIZED_DIGITAL_ASSETS** `SERIALIZED_DIGITAL_ASSETS("<property_id>")`	Returns digital asset metadata for the specified property serialized as JSON	`SERIALIZED_DIGITAL_ASSETS("Main Image")` {:"salsify:id"=>"c232473f74f6e16ed1d239bf6cf5cf50b3e25173", :"salsify:name"=>"shutterstock_380042722", :"salsify:created_at"=>Sun, 28 Feb 2016 00:11:30 UTC +00:00, :"salsify:updated_at"=>Thu, 25 Jan 2018 18:09:43 UTC +00:00, :"salsify:status"=>:completed, :"salsify:asset_height"=>2579, :"salsify:asset_width"=>3869, :"salsify:asset_resource_type"=>"image", :"salsify:filename"=>"shutterstock_380042722.jpg", :"salsify:bytes"=>2768247, :"salsify:format"=>"jpg", :"salsify:etag"=>"9fc062b0af4baa3d36d1bbbd519aecc9", :"salsify:system_id"=>SalsifyUuid(s-5941e09e-f8ce-4820-a7b9-bdbf6f13d45e), "Manufacturer"=>"Wildflower Imports, Inc."}
Arrays & Advanced Formulas	**SLICE** `SLICE("<propertyID>", "<N_start_position>", "<N_values_to_return>")`	SLICE will allow you to take a section/subset of an array. The second argument is the first value to return and the third (optional) is the number of	`SLICE(VALUES("Features"), 2, 2)` Where *Features* has 4 values: 10%

Formula Type	Formula & Syntax	Definition	Example & Output
Combining & Transforming Values	**SPLIT** `SPLIT("<propertyID1>", "<delimiting_character>")`	Separates a single value into multiple values, using a specified charcter as the separator/delimiter.	`SPLIT(VALUE("Bullets"), ",")` Where *Bullets* is 1 Year Warranty,Made in USA,80% post-consumer recycled materials, output is: 1 Year Warranty Made in USA 80% post-consumer recycled materials
Working with Numbers	**SQUARE_ROOT** `SQUARE_ROOT("<propertyID>")`	Calculates the square root of the specified value. Accepts values stored in properties and string values.	`SQUARE_ROOT(VALUE("Package Width"))` Where value for *Package Width* is 64, output is 8.
Combining & Transforming Values	**SQUISH** `SQUISH("<propertyID>")`	Outputs value with all leading and trailing whitespace removed, and any extra whitespace inside the value condensed to a single space.	`SQUISH(VALUE("Description"))` Where value for *Description* is " This is the description. ", output is: "This is the description."
Combining & Transforming Values	**STRIP_HTML** `STRIP_HTML("<propertyID>")`	Removes all HTML markup and leaves just the regular text.	`STRIP_HTML(VALUE("Description"))` Where *Description* is **My New Product** This product fixes all ills!... ", output is My New Product This product fixes all ills!...

Combining & Transforming Values	**SUBSTITUTE**
	`SUBSTITUTE("<propertyID>", "<find text>", "<replace_text>")`
		Searches for a string in a value and replaces it with what's defined. Can do multiple subsitutions with the same formula by adding comma-separated pairs of find, replace strings.	`SUBSTITUTE(VALUE("Product Description"), "•", "•")`
			`SUBSTITUTE(VALUE("Product Name"), "®", "", "™", "")`
			
			Where *Product Description* is • Durable and Lightweight, output is
			• Durable and Lightweight
			
			Where Product Description is • Durable and Lightweight with Velcro™ closure , output is
			• Durable and Lightweight with Velcro closure

Combining & Transforming Values	**SUBSTRING** `SUBSTRING("<propertyID>", <N character>, <string_length>)`	Similar to MID in Excel. Outputs a substring of a value starting at the Nth character and continuing for length of characters specified.	`PROPER(SUBSTRING(VALUE("Feature Bullet"),9,23))` Where the value for *Brand* is Features padded shoulder straps and zippered pouch, output is Padded Shoulder Straps.
Combining & Transforming Values	**SUBSTITUTE_VALUE** `SUBSTITUTE_VALUE("<propertyID>", "<find_text>", "<replace_text>")`	Searches for a string in a value and replaces it with what's defined in a value pair of find and replace text. Unlike `SUBSTITUTE`, it cannot handle multiple value pairs.	`SUBSTITUTE_VALUE("Product Description", "•", "")` Where *Product Description* is • Durable and Lightweight, output is Durable and Lightweight
Working with Numbers	**SUBTRACT** `SUBTRACT("<propertyID>", "<propertyID *or* numeric value>")`	Arithmetic function applied to numeric property values or numbers to obtain calculated values. Accepts 2 values.	`SUBTRACT(VALUE("MSRP"),"2")` Where *MSRP* is 19.99, output is 17.99
Combining & Transforming Values	**TEXT** `TEXT("<propertyID>", "<format>")`	Reformats numeric values to values stored as text, with specified format. Include decimal even when returning	`TEXT(VALUE("MSRP"), "0.00")`

Formula Type	Formula & Syntax	Definition	Example & Output
		only whole numbers (ie. to return two digit places like 01, format should be 00.)	Where value for *MSRP* is 19.9999 output is 19.99.
Pulling Out Values	**TODAY** `TODAY()`	Returns today's date. Used in combination with date information to evaluate IF statements. Useful for date-based product status.	`IF(GT(VALUE("Active Date"),TODAY()),"Pending", "Active")` Where *Active Date* is today or in the future, output is Pending. Where active date is in the past, output is Active.

Digital Asset URLs & Metadata	**TRANSFORM_ASSET_FORMAT**
	`TRANSFORM_ASSET_FORMAT("<propertyID or string>", "<format>")`
		Changes the file extension for a Salsify digital asset URL to specified format. Use in combination with TRANSFORM_ASSET_URL(s) formulas to apply tranformations that change file dimensions or other digital asset attributes. Can be applied to an array of digital asset URLs and an optional index argument.	`TRANSFORM_ASSET_FORMAT("Main Image", "png")`
			
			Where *Main Image* contains http://salsify.com/image.jpg, output is http://salsify.com/image.png.
			
			Array example:
			
			`TRANSFORM_ASSET_FORMAT(VALUES("Additional Product Images"),"png")`
			
			Where *Additional Product Images* contains
			http://salsify.com/image1.jpg
			http://salsify.com/image2.jpg
			http://salsify.com/image3.jpg
			
			Output is
			http://salsify.com/image1.png
			http://salsify.com/image2.png
			http://salsify.com/image3.png
			
			Optional index example:
			`TRANSFORM_ASSET_FORMAT("Additional Product Images","png",2)`
			
			Where *Additional Product Images* contains
			http://salsify.com/image1.jpg
			http://salsify.com/image2.jpg
			http://salsify.com/image3.jpg
			
			Output is
			http://salsify.com/image2.png

Digital Asset URLs & Metadata	**TRANSFORM_ASSET_URL** `TRANSFORM_ASSET_URL("<propertyID or string>", "<transformation_string>","<optional_index>")`	Inserts transformation string in a single Salsify URL to change the size or other characteristics of an image. See Transforming Image Files for available transformations. To tranform multiple assets, see TRANSFORM_ASSET_URLS. To also transform file type/format, see TRANSFORM_ASSET_FORMAT.	`TRANSFORM_ASSET_URL("Main Image", "c_fit,w_2000,h_2000",2)` Where *Main Image* is http://salsify.com/image.jpg, output is http://salsify.com/image/c_fit,w_2000,h_2000.jpg, transforming the image to fit within 2000x2000, maintaining original proportions.
Digital Asset URLs & Metadata	**TRANSFORM_ASSET_URLS** `TRANSFORM_ASSET_URLS("<propertyID or string>", "<transformation_string>","<optional_index>")`	Inserts transformation string for one or more Salsify URLs to change the size or other characteristics of an image. See Transforming Image Files for available transformations. To also transform file type/format, see TRANSFORM_ASSET_FORMAT.	`TRANSFORM_ASSET_URLS("Additional Product Images", "c_fit,w_2000,h_2000",2)` Where *Additional Product Images* contains http://salsify.com/image.jpg and http://salsify.com/image2.jpg output is http://salsify.com/image/c_fit,w_2000,h_2000.jpg, transforming the images to fit within 2000x2000 and http://salsify.com/image2/c_fit,w_2000,h_2000.jpg, transforming the images to fit within 2000x2000, maintaining original proportions.

Formula Type	Formula & Syntax	Definition	Example & Output
Arrays & Advanced Formulas	**UNIQ** `UNIQ(["value1","value2","value 3"...])`	Removes duplicate values in value array. Accepts an array of arrays, It will take an array of arrays, but only de-dupes the full array against another full array.	`UNIQ(["Red","Green","Red"])` Red Green
Pulling Out Values	**UPDATED_AT** `UPDATED_AT()`	Returns the date of most recent product record update to any property, in UTC, format yyyy-mm-dd.	`UPDATED_AT()` Where record was last updated on April 24, 2018, output is 2018-04-24.
Combining & Transforming Values	**UPPER** `UPPER("<propertyID>")`	Converts the given value into all uppercase.	`UPPER(VALUE("Brand"))` Where value for *Brand* is Acme, output is ACME
Pulling Out Values	**VALUE** `VALUE("<propertyID>", <optional_N>)`	Outputs the value of the given property. If the property has multiple values, it returns only one. Specify a number for N to return the value from that position.	`VALUE("Brand")` Where value for *Brand* is Acme, output is Acme.
Arrays & Advanced Formulas	**VALUES** `VALUES("<propertyID>", <optional_index>, >)`	For use with array formulas where multiple values can be returned for single properties. Use the index to indicate the first value to be returned, and length to indicate how many values should be returned total.	`VALUES("Features", 2)` Where an array contains values: Blue, White, Red, output is: White Red `VALUES("Features",1,2)` would output values: Blue White
Pulling Out Values	**WORD_COUNT** `WORD_COUNT("<string to count occurrences of>","<string to search in>")`	Counts the number of times one word or phrase appears within a text string. Does not accept arrays. Not case sensitive.	`WORD_COUNT("love",VALUE("Description"))` Where property ID *Description* contains the value You're going to love, love, LOVE this product!, output is `3`.

Formula Guidelines & Best Practices

There are a few basics to keep in mind when using Salsify formulas:

Formulas use Property IDs, not Property Names

In most cases, the Property ID and Property Name will be the same, but in some situations they're different. Use the Property ID when building formulas. Not sure if the name is the same as the ID? Check by going to the More menu, select Properties, and do a search for your property.

Formula property references are case sensitive

If the property ID is "Brand", "BRAND" and "brand" will not work.

Formula functions are not case sensitive

`VALUE` will work the same as `value` - best practice is to use all caps to make the formula easier to read.

Use straight quotation marks

In some cases, if you're using word processing software like Word to write your formulas, it inserts curly quotation marks rather than straight. If you have a formula that is unexpectedly not working, check the quotation marks to be sure they're the right type.

Use single or double quotation marks

In normal formulas, double and single quotes can be used interchangeably.

Quotes around numeric values

In most cases, you do not have to use quotes around numbers when they are a formula function like a position or whole number. But you can use quotes in these cases as well. Use quotation marks around numbers when they are being used as a value or have a decimal point in them. When in doubt, check the table above for syntax.

The SALSIFY_ prefix

When using formulas in Excel formatted templates, use `SALSIFY_` in front of every Salsify formula. So for example, when using `VALUE` in an Excel template, enter it as `SALSIFY_VALUE`. For more help with setting up Excel formatted templates, click here.

NULL vs ""

Use `NULL` as the last argument for the case where you want to insert “no value”. Use `""` where you want to insert a blank value. The readiness report interprets `NULL` as “fill in nothing”, and `""` as “fill in a blank value”. So `NULL` doesn’t get counted as a value in the completed percentage, and `""` gets counted as a value.

Special characters

There are some situations where special characters like \ or a double quotation mark need to be “escaped” in order for the formula to interpret them correctly. See *Escape Special Characters in Formulas* for more information about escaping special characters.

Formula Editing & Troubleshooting

Parentheses and quotation marks always come in pairs

When writing and troubleshooting formulas, check where your quotation marks and parentheses are, and whether they are all in pairs. If your quotation marks and parentheses are either misplaced or don’t close properly, your formula will not validate.

Look for the red underline in the readiness report formula builder

The formula builder will give you hints about where your formula isn’t validating. If there’s a red underline under a specific part of your formula like `VALUE`, check the remainder of that section to be sure that the property ID is correct and that your syntax is right.

Use formula builder to find the right property ID

An easy trick to finding the right version of your property ID is to start typing it in the bar at the top of the formula builder. Don’t select it, but the formula builder has a list of all the valid property IDs.

Formula builder can help with templated export troubleshooting

If you’re working on a templated export and you can’t figure out why your formula won’t validate, try opening up a readiness report in another window, and use one of the source fields. Paste your formula in the formula builder, remove `SALSIFY_` and see where the error is. It can be a really quick way to see a syntax error or mistake in property ID. It’s often helpful to use the formula builder like a scratch pad, validating your formulas before you add them to your template. Just remember to add `SALSIFY_` to each of your formula pieces when you add it to the Excel template.

Use line breaks in formula builder to make formulas easier to read

Splitting up a formula makes it a lot easier to both read to figure out what it does, and troubleshoot to make sure you have your parentheses and quotation marks where they belong. The formula builder ignores line breaks in formulas, so split them up

however you'd like. Here's an example:

Single line version:

```
IF(AND(VALUE("Brand"),VALUE("Category")),CONCATENATE(VALUE("Brand"," - ",VALUE("Category")),NULL)
```

Multi-line version:

```
  IF(
AND(
  VALUE("Brand"),
  VALUE("Category")
  ),
   CONCATENATE(
        VALUE("Brand"),
        " - ",
        VALUE("Category")
        ),
   NULL
  )
```

The multi-line version makes it much easier to see where everything should be, and that you have all your syntax correct.

Comment your formulas to remember what they do

You can add comments to your formulas to help yourself remember what they do, and to inform others who might be viewing or editing your formulas. Start comment lines with a # and anything on the line will be ignored, or put the comment on a line after a formula by adding the # after the formula part ends. For our example above, we could add comments like these:

```
#you can comment out an entire line by starting with #, or end a line by adding a #comment behind the formula as belo
w:
IF(
 AND(
  VALUE("Brand"),
  VALUE("Category") #if the product has both brand and category filled
  ),
   CONCATENATE(
        VALUE("Brand"),
        " - ",
        VALUE("Category") #output will be Brand - Category
        ),
   NULL  #otherwise no output at all
  )
```

Everything after the # on each line will be ignored by the formula builder.