Capture Group References

In Wrangle transformations that support use of patterns, you may need to specify capture groups. A **capture group** is a pattern that describes a set of one or more characters that constitute a match. These matches can be programmatically referenced in replacement values.

- These patterns are described using regular expression syntax. Trifacta implements a version of regular expressions based off of *RE2* and *PCRE* regular expressions.

Basic Capture Groups

Example 1

```
replace col:* on:`{start}(%+) ` with:'First Word\:$1'
```

Elements of the matching pattern (on:):

Reference	Description
{start}	A Trifacta pattern reference to the start of the tested value.
(%+)	Matches on one or more characters of any time. **NOTE:** The parentheses indicate that this set of characters is a capture group.
	Last character in the matching pattern is an empty space.

Matches: First set of any characters in the tested value up to the first empty space (the first word), across all columns of the dataset.

Replaced with: The text value `First Word:` followed by a reference to the first capture group (`$1`), which returns the first word found in the tested value.

Example 2

The previous example works fine, as long as there is a space in the tested value to identify the end of the first word. If there is one and only word in the tested value, then you must amend the `on:` parameter value with the following:

```
replace col:* on:`{start}(%+) ( |{end})` with:'First Word\:$1'
```

In this case, the second capture group features two elements:

Reference	Description
	first character in the second capture group is an empty space.
|	Logical OR, which means that the capture group matches on either the empty space or the following value, which is a reference to the end of the tested value.
{end}	A Trifacta pattern reference to the end of the tested value.

Example 3

```
replace col:* on:`{start}(%+) (%+)( |{end})` with:'Second Word\:$2'
```

Matches: The `on:` pattern has been augmented to include the second word in the tested value, across all columns of the dataset.

Replaced with: The text value `Second Word:` followed by a reference to the second capture group (`$2`), which returns the second word found in the tested value.

Dollar sign in replace transform

The dollar sign ($) is used as a form of escape character in the `with` parameter of the `replace` transform. This pattern identifies the replacement string.

In the table below, you can review how these replacement patterns are supported across the running environments.

Pattern	Description	On Photon	On Spark
$$	Inserts a $ in the replacement value.	Yes	Yes
$n or $nn	For non-negative digits n, this pattern inserts the nth parameterized sub-match string, provided that the first argument was a regex object.	Yes	Yes

Examples

In the following example, the `MyColumn` column contains the value `foobar` in all rows.

source value	`replace` transform	replacement
`foobar`	`replace col:MyColumn with:'$$f' on:'f'`	`$foobar`
`foobar`	`replace col:MyColumn with:'$2' on:`(f)(o)(o)bar`` Note that the `on` parameter is a Trifacta® pattern.	`o`

Positive and Negative Lookaheads

In regular expressions, you can use positive and negative lookahead capture groups to capture content that is conditionally followed or not followed by a specified capture group.

Type	Example expression	
Positive lookahead	`/q(?u)/`	Capture the letter q only when it is followed by the letter u. Letter u is not captured.
Negative lookahead	`/q(?!u)/`	Capture the letter q when it is not followed by the letter u. Letter u is not captured.