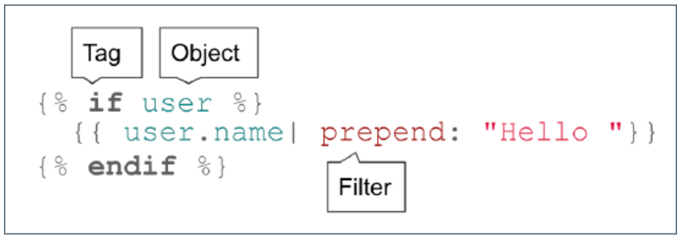
Use Liquid expressions in Workflow

Liquid is an open-source template language created by Shopify. Liquid uses a combination of objects, tags, and filters inside template files to display dynamic content.

- Object: Contain the content that Liquid displays on a page. To output an object's property, wrap the object name in `{{` and `}}`.
- Tag: Create the logic and control flow for templates. Tags are wrapped in: `{% %}` characters. An object inside a tag does not need brackets.
- Filter: Modify the output of a Liquid object or variable. Filters are placed within an output tag `{{ }}` and are denoted by a pipe character `|`.

Here is an example. For details about objects, filters, tags, and other tips for Liquid, see Liquid basics and Liquid reference.



```
{% if user %}
  {{ user.name| prepend: "Hello "}}
{% endif %}
```

You can use Liquid expressions in Zuora Workflow tasks, including the task name, to filter and transform data. The following examples provided in this article can be used as references. The sample outputs are provided in the comments `{% comment %} {% endcomment %}`.

The version of Liquid used in Workflow is 5.0.1.

Filters for date and time

date

Convert a timestamp to another date format.

Parameters: input, date format

> {{ "now" | date: "%Y-%m-%d" }}
> {% comment %} 2021-06-11 {% endcomment %}

```
{{ "now" | date: "%FT%T%:z" }}
{% comment %} 2021-06-11T02:00:00 {% endcomment %}
```

date_manip

Add or subtract time from a date.

Parameters: input, operation['-', '+'], an integer indicating the number of time to be added/subtracted, metric['minute', 'hour', 'day', 'month', 'year']

```
{{ '2021-01-01' | date_manip: '+', 1, 'day' | date: '%Y-%m-%d' }}
{% comment %} 2021-01-02 {% endcomment %}
```

date_between

Check if the input is between date parameters. Both dates are inclusive.

Parameters: input, start date, end date

```
{{ '2021-01-01' | date_between: '2000-01-01', '2050-01-01'}}
{% comment %} true {% endcomment %}
```

date_diff

Calculate the number of days between parameters.

Parameters: input, date2

```
{{ '2021-01-01' | date_diff: '2021-01-08'}}
{% comment %} 7 {% endcomment %}
```

timezone

Convert the date and time to a specific time zone.

Parameters: input, previous format, current format, timezone

```
{% assign dateTimeFormat = '%Y-%m-%d %H:%M %Z' %}
{{Data.Workflow.ExecutionDateTimeUTC | date: dateTimeFormat | timezone: dateTimeFormat,
dateTimeFormat, 'America/New_York'}}
{% comment %} 2021-01-01 03:00 EST {% endcomment %}
```

http_date

Convert the date and time to the HTTP-date format.

Parameters: input

> {{ "now" | http_date }}
> {% comment %} Tue, 11 Jul 2017 06:00:00 GMT {% endcomment %}

in_time_zone

Convert a non-UNIX time to a specific time zone.

Parameters: input, timezone

> {{ 'now' | in_time_zone: 'America/New_York' }}
> {% comment %} 2021-01-01 03:00:00 -0300 {% endcomment %}

Filters for array manipulation

pop

Remove elements from the end of the array.

Parameters: input, an integer indicating the number of elements to be removed (1 by default)

> {{'a,b,c' | split: ',' | pop }}
> {% comment %} ['a','b'] {% endcomment %}
> {{'a,b,c' | split: ',' | pop: 2 }}
> {% comment %} ['a'] {% endcomment %}

push

Add an element to the end of the array.

Parameters: input, the element to be added

> {{'a,b,c' | split: ',' | push: 'd' }}
> {% comment %} ['a','b','c','d'] {% endcomment %}

shift

Remove elements from the start of the array.

Parameters: input, an integer indicating the number of elements to be removed (1 by default)

> {{'a,b,c' | split: ',' | shift }}
> {% comment %} ['b','c'] {% endcomment %}

unshift

Add an element to the start of the array.

Parameters: input, the element to be added

> {{'a,b,c' | split: ',' | unshift: 'z' }}
> {% comment %} ['z','a','b','c'] {% endcomment %}

where

Select all the objects in an array where the key has the given value.

Parameters: input, property, value

> {{ Data.Invoice | where: 'AccountId', '2c92c0fb72a8521c0172a96f55581405' }}
> {% comment %}
> [{"AccountId"=>"2c92c0fb72a8521c0172a96f55581405","InvoiceNumber"=>"INV00000001","Balance"=>10}] {% endcomment %}

where_exp

Select all the objects in an array where the expression is true.

Parameters: input, variable, expression

Note: The and/or operators are not currently supported in the expression, though you can replicate "and" with chained `where_exp` filters and "or" with the `concat` filter.

> {{ Data.Invoice | where_exp: 'item', 'item.Balance > 100' }}
> {% comment %}
> [{"AccountId"=>"2c92c0fb72a8521c0172a96f55581405","InvoiceNumber"=>"INV00000002","Balance"=>115}]
> {% endcomment %}

group_by

Group the items in an array by a given property.

Parameters: input, property

> {{ Data.Invoice | group_by: 'Status' }}
> {% comment %}
> [{"name"=>"Draft","items"=>[{"Status"=>"Draft","InvoiceNumber"=>"INV00000003"}],"size"=>1},{"name"=>"Posted","items"=>
> {% endcomment %}

group_by_exp

Group the items in an array using a Liquid expression.

Parameters: input, variable, expression

> {{ Data.Invoice | group_by_exp: 'item', 'item.InvoiceDate | date: "%m"' }}
> {% comment %}
> [{"name"=>"01","items"=>[{"InvoiceDate"=>"01","InvoiceNumber"=>"INV00000004"},{"InvoiceDate"=>"01","InvoiceNumber"=>
> "size"=> 2},{"name"=>"02","items"=>[{"InvoiceDate"=>"02","InvoiceNumber"=>"INV00000006"}],"size"=>1}]{%
> endcomment %}

map and join

> RatePlanChargeId = '{{ Data.Usage | map: 'ChargeId' | join: "' OR RatePlanChargeId = '"}}'
>
> ProductId = '{{ Data.Product | map: 'Id' | join: "' AND Region__c = 'US' OR ProductId = '"}}' AND Region__c =
> 'US'

- You can use `map` and `join` to query for an object using an array of records.
- `map` lists the array values of `ChargeId` on `Data.Usage`. `join` puts the array in a single string with `OR RatePlanChargeId` in between each value.
- There is a single space before `OR`.
- If you need to include another operator, you can add `' AND [Field Name] = ''` before `OR` and close with one last `AND` condition.

String Utilities

to_json

Convert the format from hash to JSON.

> {{ Data.Workflow | to_json }}
> {% comment %} {"ExecutionDate":"2021-01-01","LastRunDate":"2020-12-31"...} {% endcomment %}

parse_json

Convert the format from JSON to hash.

```
{% assign input = '{"name":"INV-100","balance":1}' | parse_json %}
{{ input }}
{% comment %} {"name"=>"INV-100","balance"=>1} {% endcomment %}
```

string_escape

Escape special characters.

```
{{ '\abc\' | string_escape }}
{% comment %} \\abc\\ {% endcomment %}
```

to_xml

Convert the format from hash to XML.

```
{{ Data.Workflow | to_xml}}
{% comment %} <?xml version="1.0" encoding="UTF-8"?><hash><ExecutionDate>2021-01-
01</ExecutionDate>...</hash> {% endcomment %}
```

regex

Return array of strings that match the regular expression.

Parameters: input, regular expression, operation['match_first', 'match_all']

```
{{ 'INV-100,INV-101' | regex: 'INV-\d+', 'match_first' }}
{% comment %} ["INV-100"] {% endcomment %}

{{ 'INV-100,INV-101' | regex: 'INV-\d+', 'match_all' }}
{% comment %} ["INV-100","INV-101"] {% endcomment %}
```

input_byte_8x

Add spaces to a string to reach the byte mark.

```
{{ 'abc' | input_byte_8x }}
{% comment %} 'abc     ' {% endcomment %}
```

Filters for Zuora utilities

check_processed_tag

Check if any previous run of this task has the same tag logged.

Parameters: tag, original task ID

Note: It only checks runs still in history which will expire later.

> {{ 'INV-100' | check_processed_tag: TaskInstance.original_task_id }}
> {% comment %} true {% endcomment %}

check_parent_workflow_status

Check if any workflow runs with the specified name/ID are in the given status.

Parameters: original workflow id or name, status['Queued', 'Processing', 'Stopped', 'Stopping', 'Finished']

> {{ WorkflowInstance.id | check_parent_workflow_status: 'Queued' }}
> {% comment %} true {% endcomment %}
>
> {{ WorkflowInstance.name | check_parent_workflow_status: 'Processing' }}
> {% comment %} false {% endcomment %}

is_ar_enabled

Return true if Invoice Settlement is enabled in the Zuora tenant.

> {{ nil | is_ar_enabled }}
> {% comment %} true {% endcomment %}

Zuora helpers

> {{ Credentials.zuora.username }}
> => The Username of the Workflow instance

> {{ Credentials.zuora.password }}
> => The Password of the Workflow instance

> {{ Credentials.zuora.url }}
> => The SOAP API URL of the Workflow instance

> {{ Credentials.zuora.session }}
> => The Workflow instance session

> {{ Credentials.zuora.entity_id }}
> => The Entity Id of the Workflow instance

> {{ Credentials.zuora.rest_endpoint }}
> => The REST API URL of the Workflow instance. For example, *https://rest.zuora.com/v1/*.

Encoding

You can use the following filters for encoding:

- md5: Convert to MD5.
- sha1: Convert to SHA1.
- sha2: Convert to SHA256/SHA384/SHA512.
- sha256_encode64: Convert to SHA256 with base 64.
- base64_encode: Encode in base 64.
- base64_decode: Decode in base 64.
- unpack: Convert to hex.
- pack: Convert from hex.
- utf_8_encoding: Convert to UTF-8 encoding.

```
{{ 'abc' | md5 }}
{% comment %} 900150983cd24fb0d6963f7d28e17f72 {% endcomment %}

{{ 'abc' | sha1 }}
{% comment %} a9993e364706816aba3e25717850c26c9cd0d89d {% endcomment %}

{{ 'abc' | sha2 }}
{% comment %}
ddaf35a193617abacc417349ae20413112e6fa4e89a97ea20a9eeee64b55d39a2192992a274fc1a836ba3c23a3feebbd454d44
{% endcomment %}
{{ 'abc' | sha2: 'SHA256' }}
{% comment %} ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad {% endcomment
%}

{{ 'abc' | sha256_encode64 }}
{% comment %} ungWv48Bz+pBQUDeXa4iI7ADYaOWF3qctBD/YfIAFa0= {% endcomment %}

{{ 'abc' | base64_encode }}
{% comment %} YWJj {% endcomment %}

{{ 'YWJj' | base64_decode }}
{% comment %} abc {% endcomment %}

{{ 'abc' | unpack }}
{% comment %} 616263 {% endcomment %}

{{ '616263' | pack }}
{% comment %} abc {% endcomment %}
```

Encryption

rsa_encrypt

Encrypt with RSA.

Parameters: data, action['encrypt', 'decrypt'], key

```
{% assign public_key = '-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCdlUof3WUjlLwlwva420D5nnXh
xwOqYAUzHIq/AUzBYkDDla2hTtvspcebiFnKtXWmlZrzu4dNYtvE/2ZPiqrmPl8l
PZAX74WwDRjMCC8HPGUmt6/2mVtXQ7wM9R1KEe9kX0lLeRLmK+lW24KvSL8U1yat
WO1TP4GmTZ7Kw21fDQIDAQAB
-----END PUBLIC KEY-----' %}
{% assign input = 'encryptedMessage' %}
{{ input | rsa_encrypt: 'encrypt', public_key }}
{% comment %} Wha580r+8bO/D2dBXS1zxkzQKnJZpR3EGu8h4a0hx5QPhY7GUJVTU5le/
D5bRQDyiWlckz1hMPIAj3rU79eF04d3XLKCgD07XKoIEkQ1e3/
i3s2FWhFyhyIPgOeqdwF9M89nx62epoLGGiKmxew+OnWTyRXwfIB8BRx16vTD9wY= {% endcomment %}
```

rsa_decrypt

Decrypt with RSA.

Parameters: input, private key

```
{% assign private_key = '-----BEGIN RSA PRIVATE KEY----- MIICXQ
IBAAKBgQCdlUof3WUjlLwlwva420D5nnXhxwOqYAUzHIq/AUzBYkDDla2h
TtvspcebiFnKtXWmlZrzu4dNYtvE/2ZPiqrmPl8lPZAX74WwDRjMCC8HPGUmt6/2
mVtXQ7wM9R1KEe9kX0lLeRLmK+lW24KvSL8U1yatWO1TP4GmTZ7Kw21fDQIDAQAB
AoGAQCRoS5geduEvxF8bdhso03JAoWoUf+EdvLQ9dYnd6ElJ+1KNnj8vHaBNI23Y
vr4l6Wyz6cnHRSScOA+NYYscDICJKbVvYR4CMezjkaNCPwzEezg0ievNHZG3jaSG
SALVbOA3ZR3zjZoLILYbuXeNEOVhyQvTcXQuiDWJDF9okTECQQDLQeFDW+9Yj/aN
nZlPrGUGVFg7DPPjKS+f+iAbxX0Dy5Dx+oENbSWQoPq8Gkrp47Ih0o+NTTBawXf/
5TPOON0XAkEAxnlVV28sL0jLM/XlTmkzF3ALNTOheatnpCtDmhUoIsczr0qrqPHD
FhzXaJGCFrRNw578OudK7zEmFpb8LccjewJBAIsLsW9kGBNMwMzWMEgJ1j6DOqyC
yuDuju7wrEBzVHdhLfHrfZdSwGz1QzGlBvSD2Js8sQln8ZlUWqQLBuqfidcCQGlW
dlv8bif+a5L9XCG8pdg3qEspRveLaGtyZuO9fNrQfPb5G5iED/WJgy5FpBSmx2se
02Hz5gqDPGLa6abmPUcCQQCFgObpLkVla0amHz/U7y7pNASK3KzRxRAKJ/RJ5pQo
rPKZkC+HMMeaq7z1ZBBZ2tRM+BFN13dF/CaUWwGe4fML
-----END RSA PRIVATE KEY-----' %}
{% assign input = 'Wha580r+8bO/D2dBXS1zxkzQKnJZpR3EGu8h4a0hx5QPhY7GUJVTU5le/
D5bRQDyiWlckz1hMPIAj3rU79eF04d3XLKCgD07XKoIEkQ1e3/
i3s2FWhFyhyIPgOeqdwF9M89nx62epoLGGiKmxew+OnWTyRXwfIB8BRx16vTD9wY=' %}
{{ input | rsa_decrypt: private_key }}
{% comment %} encryptedMessage {% endcomment %}
```

symmetric_encrypt

Encrypt/Decrypt with the specified symmetric algorithm.

Parameters: input, action['encrypt', 'decrypt'], mode, key, iv = nil, no_padding['true', 'false']

> {% assign input = 'encryptedMessage' | base64_encode %}
> {% assign iv = 'initialVector000' | base64_encode %}
> {% assign key = 'encryptionKey000' | base64_encode %}
> {{ input | symmetric_encrypt: 'encrypt', 'AES-128-CBC', key, iv, false | base64_encode}}
> {% comment %} abby2vv8nUuoiagV6P35KY2WKrSQHQ50ihDzRE0uvjI= {% endcomment %}

aes_decrypt

Decrypt with AES.

Parameters: input, decryption_IV, decryption key, mode['AES-128-CBC', 'AES-256-CBC']

> {% assign input = 'abby2vv8nUuoiagV6P35KY2WKrSQHQ50ihDzRE0uvjI=' %}
> {% assign iv = 'initialVector000' | base64_encode %}
> {% assign key = 'encryptionKey000' | base64_encode %}
> {{ input | aes_decrypt: iv, key, 'AES-128-CBC' | base64_decode }}
> {% comment %} encryptedMessage {% endcomment %}

jwt_encode/jwt_decode

Encode/Decode JWT with the specified algorithm.

Parameters for encoding: payload, key, algorithm

Parameters for decoding: token, key, verify

> {% assign key = 'jwtKey' %}
> {{ 'abc' | jwt_encode: key, 'HS256' }}
> {% comment %} eyJhbGciOiJIUzI1NiJ9.ImFiYyI.HWTAZzvzs7q4SP9GQwXkqhN_uFQ6aFCJGPuTKUFs_ew {% endcomment %}
>
> {% assign input = 'eyJhbGciOiJIUzI1NiJ9.ImFiYyI.HWTAZzvzs7q4SP9GQwXkqhN_uFQ6aFCJGPuTKUFs_ew' %}
> {% assign key = 'jwtKey' %}
> {{ input | jwt_decode: key, true }}
> {% comment %} ['abc',{"alg"=>"HS256"}] {% endcomment %}

hmac_sha256_sign/hmac_sha512_sign

Use HMAC-SHA256/HMAC-SHA512 to sign a string.

Parameters: input, key

> {% assign key = 'testHmacKey' %}
> {{ 'abc' | hmac_sha256_sign: key }}

> {% comment %} EJBqLUcQxuSwQ9UuucfSLiG9IsE2g7+AVRApdxmbOFw= {% endcomment %}

rsa_random_key

Create a random key with the specified algorithm.

Parameters: nil, mode

> {{ nil | rsa_random_key: 'AES-128-CBC' | base64_encode }}
> {% comment %} ETnoMe4SJ/jHWAtQfdaPFw== {% endcomment %}

Other filters

data_type

Return the type of the parameter.

> {{ 'abc' | data_type }}
> {% comment %} String {% endcomment %}

money

Rounds to two decimal points with an optional currency symbol.

Parameters: money, currency (optional)

> {{ 9099.000909 | money }}
> {% comment %} 9,099.00 {% endcomment %}
>
> {{ 9099.000909 | money: 'USD' }}
> {% comment %} $9,099.00 {% endcomment %}
>
> {{ Data.Account.Balance | money: Data.Account.Currency }}
> {% comment %} £582.62 {% endcomment %}

random

Return a random integer between the input parameters.

Parameters: nil, parameter1, parameter2

> {{ nil | random: 1,10 }}
> {% comment %} 8 {% endcomment %}

random_variable

Return a random UUID.

Parameters: input, type['uuid']

```
{{ nil | random_variable: 'uuid' }}
{% comment %} d6bbd069-7a24-4357-a6cc-1aec5852067d {% endcomment %}
```

Use case examples for tags

Control flow tags

if/then

```
{% if Data.Callout.currency == "USD" and Data.Callout.amount > 10 %} True
{% elsif Data.Callout.currency == 'MXP' and Data.Callout.amount >100 %} True
{% else %} False
{% endif %}
```

- When used in Workflow, `True` and `False` must be spelled with the first letter capitalized.
- `{{ }}` is not required around each field since it is already in a logic `{% %}`.

Null check

```
{% if a.ProductCatalogGroup__c != blank %}
"ProductCatalogGroup__c": "{{ a.ProductCatalogGroup__c }}",
{% endif %}
```

Evaluate the size of an array

```
{% if Data.Subscription.size > 0 %}
True
{% else %}
False
{% endif %}
```

Determine the presence of an array

```
{% if Data.ratePlans[index].lastChangeType %}
```

case

```
{% case Data.Subscription.Count_Name and Data.Account.Currency %}
{% when 1 and 'USD' %} One
{% when 2 and 'USD' %} Two
{% when 3 and 'USD' %} Three
{% endcase %}
```

Iteration tags

Iterate over JSON

In Workflow Iterate task, you may enter a JSON payload that is not in the dropdown list. When doing so, you will need to format the payload by using the to_json filter so that the Workflow recognizes that what you entered is JSON.

```
{{Data.Callout[0].ResponseBody.order.subscriptions.orderActions | to_json }}
```

for loop

```
{% for RP in Data.RatePlan %}
{{ RP.Name }} - the piece that loops
{%- if forloop.last == false -%} , {% endif %} {%- endfor %}
```

Indexing

```
{% for ii in Data.InvoiceItem %}
   {% assign index = forloop.index0 %}
   {% assign i = Data.Invoice[index] %}
   {% assign p = Data.Product[index] %}
   {% assign s = Data.Subscription[index] %}
   {% assign prp = Data.ProductRatePlan[index] %}
   {% assign prpc = Data.ProductRatePlanCharge[index] %}
{% endfor %}
```

continue and break

```
{% for RPC in Data.RatePlanCharge %}
 {% if RPC.ChargeType != “Recurring” %}
   {% continue %}
 {% else %}
   {{ RPC.Name }}
 {% endif %}
{% endfor %}
```

- `{% continue %}` causes the loop to skip the current iteration when it encounters the `continue` tag.
- `{% break %}` causes the loop to stop iterating when it encounters the `break` tag.
- You can use it as a way to do a form of the inner join between two arrays.

Math and for loop

```
{% assign optics_total = 0 %}
{% for usage in Data.Usage %}
{% assign optics_total = optics_total | plus: usage.Quantity %}
{% endfor %}
{{ optics_total }}
```

Variable tags

Assign a subset of array data

```
{%- assign removeProducts = Data.Callout.ResponseBody.order.subscriptions[0].orderActions | where: "type",
"RemoveProduct" -%}
{%- for rps in removeProducts -%}
{{rps.removeProduct.ratePlanId}}
{%- endfor -%}
```

Round-robin

```
{% assign AgentArray = "agent1, agent2, agent3, agent4" | split: ", " %}
{% assign AccountArray = "account1, account2, account3, account4, account5, account6" | split: ", " %}
{% assign AgentArraySize = AgentArray | size %}
{% for account in AccountArray %}
{% assign AgentIndex = forloop.index0  | modulo: AgentArraySize %}
{% assign Agent = AgentArray[AgentIndex] %}
    {{Agent}}
{% endfor %}
```

Fetch a value based on sorting an array of data

```
{% assign myItem =  Data.RatePlanCharge | sort:  ‘ChargedThroughDate’ | reverse  | first %}{{myItem.
ChargedThroughDate}}
```

Other resources

- Liquid Cheat Sheet
- Liquid Sandbox (useful for testing simple syntax)
- Newlines and white spaces might interfere with your Liquid logic. See the following methods to remove unnecessary spaces.
  - https://shopify.github.io/liquid/basics/whitespace/
  - https://shopify.github.io/liquid/filters/lstrip/
  - https://shopify.github.io/liquid/filters/strip_newlines/