Formula Step Types

Last Modified on 06/17/2022 5:12 am EDT

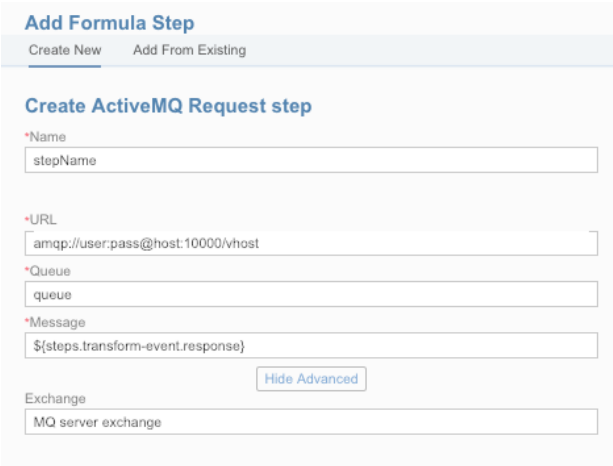
You can choose from several different types of steps to make up your formula. You can refer to any step with the `${steps.stepName}` syntax. Because you refer to the step by name, each step name must be unique within each formula. However, you can reuse a step name in a different formula. You can also add readmes and descriptions to both entire formulas and their individual steps; see Formula Readmes and Descriptions for additional information.

You can use the following types of steps in your formulas:

- ActiveMQ Request
- Connector API Request
- HTTP Request
- JS Filter
- JS Script
- Loop Over Variable
- Platform API Request
- Retry Formula on Failure
- Stream
- Sub-formula

ActiveMQ Request

The ActiveMQ Request (`amqpRequest`) step type uses the AMQP protocol to post a message to an MQ server such as RabbitMQ.




```
{
 "steps":[
  {
   "name":"stepName",
   "onFailure":[

   ],
   "type":"amqpRequest",
   "properties":{
    "exchange":"MQ server exchange",
    "body":"${steps.transform-event.response}",
    "queue":"queue",
    "url":"amqp://user:pass@host:10000/vhost"
   },
   "onSuccess":[
    "nextStepName"
   ]
  }
 ]
}
```

When you set up an ActiveMQ Request step, include the following information:

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y
URL `url`	Specifies the AMQP URL endpoint of the MQ Server. The structure of the URL is specified in RabbitMQ URI Specification.	Y
Queue `queue`	Indicates the name of the queue of the MQ server to which the message should be posted.	Y
Message `body`	The JSON payload to post to the server.	Y
Exchange `exchange`	The name of the MQ server exchange to which the message should be posted.	N

ActiveMQ Request Step Scope

ActiveMQ Request steps add the step execution values described in the example JSON below to the formula context. The formula context is then passed from step-to-step, allowing you to use these values in any subsequent steps in your formula.

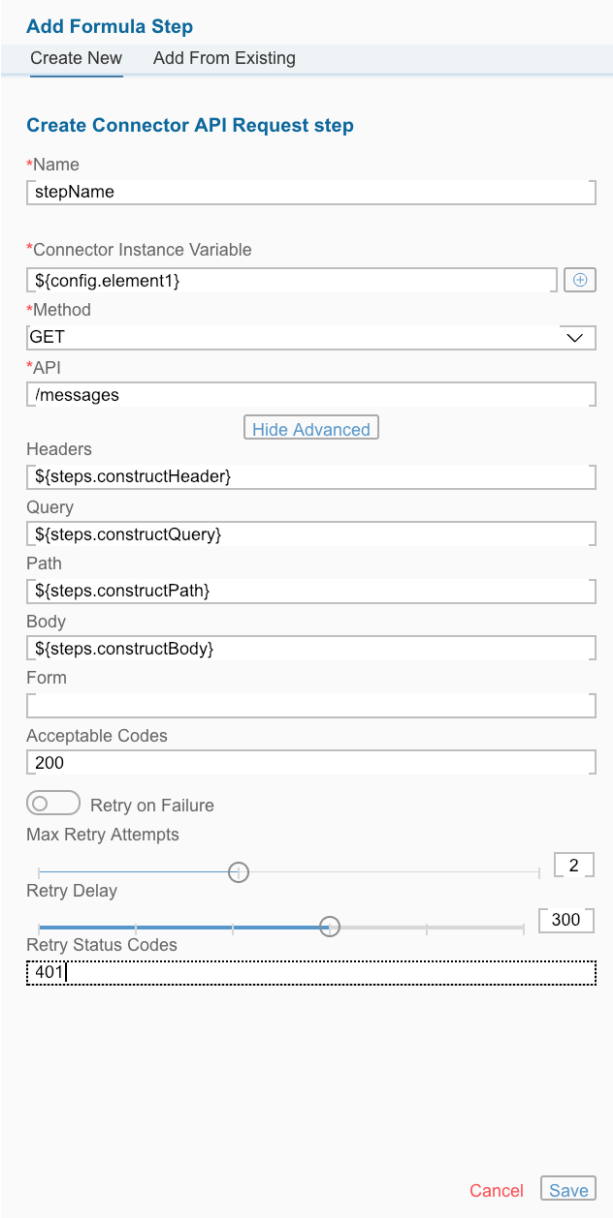
```
{
 "myAmqStep": {
  "request": {
    "body": "{\"message\":\"This is a test message.\"}",
    "url": "amqp://otqaqsml:tPpXwTl7-iMtezRmyJmD-y2U_XbroYpW@jaguar.rmq.cloudamqp.com/otqaqsml",
    "exchange": "main",
    "queue": "myqueue"
  }
 }
}
```

Example references to ActiveMQ Request scope:

- `${steps.myAmqStep.request}`
- `${steps.myAmqStep.request.body}`

Connector API Request

The Connector API Request (`elementRequest`) step makes an API call to a specific Connector Instance.




```
{
 "steps":[
  {
   "name":"stepName",
   "onFailure":[

   ],
   "type":"elementRequest",
   "properties":{
    "elementInstanceId":"${config.elementVariable
}",
    "method":"POST",
    "api":"/messages",
    "headers":"Header content",
    "query":"query string",
    "path":"path string",
    "body":"Body content",
    "acceptableStatusCodes":"200,201",
    "retry":"true",
    "retryAttempts":"5",
    "retryDelay":"401",
    "retryStatusCodes":"500,501"
   },
   "onSuccess":[
    "nextStepName"
   ]
  }
 ]
}
```

To see an Connector API Request step in action see:

- CRM to Messages
- Add New Contact Created in One System to Another
- Bulk Transfer CRM Data

When you set up an Connector API Request step, include the following information:

Parameter	Description	Required

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y
Connector Instance Variable `elementInstanceId`	Specifies the connector instance that receives the API call.	Y
Method `method`	The API method of the API call, such as GET, POST, PUT, PATCH, or DELETE.	Y
API `api`	The endpoint, such as `hubs/crm/contacts`.	Y
Headers `headers`	The headers to pass along as part of the API request. You rarely need to add anything to the headers, but you can use this parameter to pass common header information such as content types.	N
Query `query`	Specifies the filter query to send with the related request. Construct the query in another step and refer to it in the query field. For example, ${steps.previousStep.query}	N
Path `path`	Support earlier formulas where `path` defined variables, such as an {ID} variable in an endpoint. In the latest version, the path parameter is unnecessary.	N
Body `body`	Specifies the JSON body to send with the related request. Construct the JSON body in another step and refer to it in the `body` parameter. For example, `${steps.previousStep.body}`.	N
Acceptable Codes `acceptableStatusCodes`	A comma-separated list (`200,201`) of codes, range (`200-205`), or both (`200-205,208`) returned in the response that indicates success.	N
Retry on Failure `retry`	Indicates that we should retry a configurable number of times if the request fails.	N
Max Retry Attempts `retryAttempts`	The maximum number of times to retry the request.	N
Retry Delay `retryDelay`	The time in milliseconds to wait between retries.	N
Retry Status Codes `retryStatusCodes`	A comma-separated list (`500,502`) of codes, range (`400-415`), or both (`400-415,500,502`) returned in the response that indicates that we should retry the request.	N

Connector API Request Step Scope

Connector API Request steps add the step execution values described in the example JSON below to the formula context. The formula context is then passed from step-to-step, allowing you to use these values in any subsequent steps in your formula.

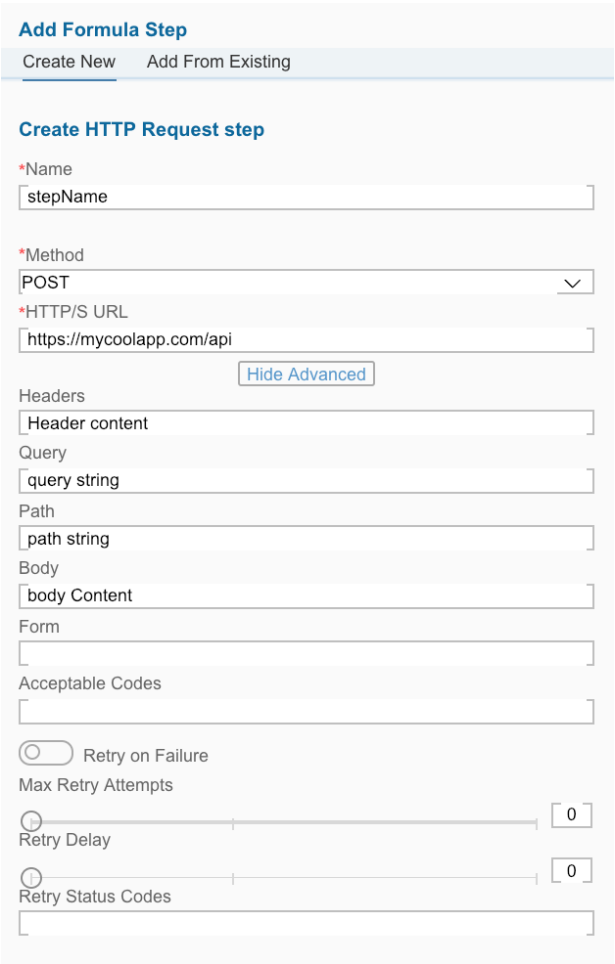
```
{
  "myElementRequestStep": {
    "request": {
      "query": "{}",
      "body": "{\"Name\":\"New Account Name\"}",
      "method": "POST",
      "path": "{}",
      "uri": "/elements/api-v2/hubs/crm/accounts",
      "headers": "{\"authorization\":\"Element /ABC=, User DEF=, Organization GHI\",\"content-
length\":\"14\",\"host\":\"jjwyse.ngrok.io\",\"content-type\":\"application/json}"
    },
    "response": {
      "code": "200",
      "headers": "{\"Set-Cookie\":
\"CESESSIONID=2CA15552EE56EAF65BF1102F6CACEACC;Path=/elements/;HttpOnly\"}",
      "body": "{\"Id\": \"001tx3WcAAI\", \"Name\": \"New Account Name\"}"
    }
  }
}
```


Example references to Connector API Request scope:

- `${steps.myElementRequestStep.request}`
- `${steps.myElementRequestStep.request.body}`
- `${steps.myElementRequestStep.response.code}`

HTTP Request

The HTTP Request (`httpRequest`) step make an HTTP/S call to any URL/endpoint.




```
{
 "steps":[
  {
   "name":"stepName",

   "onFailure":[

   ],
   "type":"httpRequest",
   "properties":{
    "method":"POST",
    "url":"https://mycoolapp.com/api",
    "headers":"Header content",
    "query":"query string",
    "path":"path string",
    "body":"Body content",
    "acceptableStatusCodes":"200,201",
    "retry":"true",
    "retryAttempts":"5",
    "retryDelay":"401",
    "retryStatusCodes":"500,501"
   },
   "onSuccess":[
    "nextStepName"
   ]
  }
 ]
}
```

When you set up an HTTP Request step, include the following information:

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y
Method `method`	The API method of the API call, such as GET, POST, PUT, PATCH, or DELETE.	Y
HTTP/S URL `url`	The full URL of the request.	Y
Headers `headers`	The headers to pass along as part of the API request. You rarely need to add anything to the headers, but you can use this parameter to pass common header information such as content types.	N
Query `query`	Specifies the filter query to send with the related request. Construct the query in another step and refer to it in the query field. For example, ${steps.previousStep.query}	N
Path `path`	Support earlier formulas where `path` defined variables, such as an {ID} variable in an endpoint. In the latest version, the path parameter is unnecessary.	N
Body `body`	Specifies the JSON body to send with the related request. Construct the JSON body in another step and refer to it in the `body` parameter. For example, `${steps.previousStep.body}` .	N
Acceptable Codes `acceptableStatusCodes`	A comma-separated list (`200,201`) of codes, range (`200-205`), or both (`200-205,208`) returned in the response that indicates success.	N
Retry on Failure `retry`	Indicates that we should retry a configurable number of times if the request fails.	N
Max Retry Attempts `retryAttempts`	The maximum number of times to retry the request.	N
Retry Delay `retryDelay`	The time in milliseconds to wait between retries.	N
Retry Status Codes `retryStatusCodes`	A comma-separated list (`500,502`) of codes, range (`400-415`), or both (`400-415,500,502`) returned in the response that indicates that we should retry the request.	N

HTTP Request Step Scope

HTTP Request steps add the step execution values described in the example JSON below to the formula context. The formula context is then passed from step-to-step, allowing you to use these values in any subsequent steps in your formula.

```
{
 "myHTTPRequestStep": {
  "request": {
     "query": "{}",
     "body": "{\"Name\":\"New Account Name\"}",
     "method": "POST",
     "url": "https://api.myservice.com:443/myresource",
     "path": "{}",
     "headers": "{\"authorization\":\"mysessionid\",\"content-type\":\"application/json}"
  },
  "response": {
   "code": "200",
   "headers": "{\"Set-Cookie\":
\"CESESSIONID=2CA15552EE56EAF65BF1102F6CACEACC;Path=/elements/;HttpOnly\"}",
    "body": "{\"id\": \"237648\", \"name\": \"My New Resource Name\"}"
  }
 }
}
```


Example references to HTTP Request scope:

- `${steps.myHTTPRequestStep.request}`
- `${steps.myHTTPRequestStep.request.body}`
- `${steps.myHTTPRequestStep.response.code}`

JS Filter

Use the JS Filter (true/false) (`filter`) step to write custom Javascript that *must* return true or false. As with all steps, you must include a name. See Javascript in Formulas for more information about working with Javascript in formulas.


```
{
 "name":"stepName",
 "onFailure":[

 ],
 "type":"filter",
 "properties":{
  "body":"Javacript, for example: let status = trigger
.args.status;\n\nif (status && status === \"COMPLET
ED\") {\n  done(true);\n} else {\n  done(false);\n}"
 },
 "onSuccess":[
  "nextStepName"
 ]
}
```


Use JS Filter steps to specify only certain event types, field values, or other information. You can also use filters to split formulas into different paths.

- If a filter returns `true` , the formula executes the left, or OnSuccess, step.

- If a filter returns `false`, the formula executes the tight, or OnFailure, step.

To see a JS Filter step in action see:

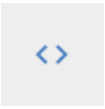
- Retrieve, Transform, and Sync Contact
- Bulk Transfer CRM Data

JS Filter Step Scope

JS Filter steps pass a boolean into the JS `done` callback function. That boolean is made available under the key titled `continue`, as shown in the examples below.

```
{
 "myFilterStep": {
  "continue": "true"
 }
}
```

```
{
 "myFilterStep": {
  "continue": "false"
 }
}
```

JS Script 

Use the JS Script (`script`) step to write custom Javascript that *must* pass a valid JSON object to the `done` callback. As with all steps, you must include a name. See Javascript in Formulas for more information about working with Javascript in formulas.


```
{
 "steps":[
  {
   "name":"stepName",
   "onFailure":[

   ],
   "type":"script",
   "properties":{
    "body":"Javacript, for example: done({\n  \"subj
ect\": \"CRM Event Occurred\",\n  \"to\": \"recipient@
gmail.com\",\n  \"from\": \"sender@cloud-elements.c
om\",\n  \"message\": `${trigger.event.objectType}
with ID ${trigger.event.objectId} was ${trigger.even
t.eventType}`\n});"
   },
   "onSuccess":[
    "nextStepName"
   ]
  }
 ]
}
```

Use JS Script steps to build objects to use in request steps for query parameters or the request body.

> **Note:** If you use `console.log` in a JS Script step, the output is added to the body of the step. If you reference the script step in another step as just `${steps.stepName}`, the `console.log` output is added to the step context and can cause errors. Prevent this by declaring what to include in the step body by adding it to `done`. For example, `done({body.variableName})`.

To see a JS Script step in action see:

- CRM to Messages
- Bulk Transfer CRM Data

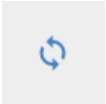
JS Script Step Scope

JS Script steps add whatever object is passed to the JS `done` callback to the formula context. The formula context is then passed from step-to-step, allowing you to use these values in any subsequent steps in your formula.

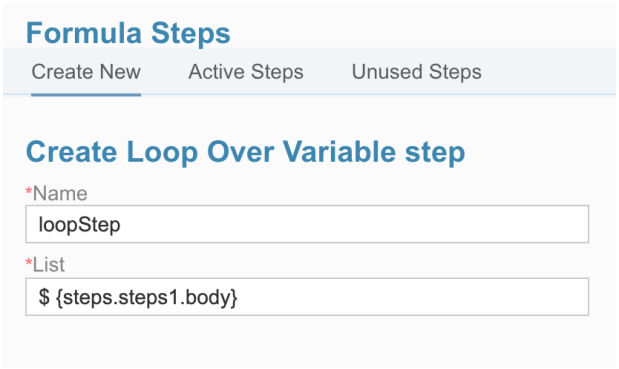
```
done({
  foo: 'bar',
  object: {
    someKey: 'someValue'
  }
});
```

Example references to JS Script scope:

- `${steps.myScriptStep.foo}`
- `${steps.myScriptStep.object}`
- `${steps.myScriptStep.object.someKey}`

Loop Over Variable 

Use the Loop Over Variable (`loop`) step to loop over a list of objects from a previous step or trigger. Set `onSuccess` to the first step in the loop. When you have reached the last step in the loop set the onSuccess field to the loop step, this will restart the loop for the next object. If you need to continue on after the loop is completed, set `onFailure` to the next step to execute after the loop is completed. For a loop step, `onFailure` is executed when the loop has been executed for all objects in the list.




```
{
  "steps":[
    {
      "name":"stepName",
      "onFailure":[

      ],
      "type":"loop",
      "properties":{
        "list":"$ {steps.steps1.body}"
      },
      "onSuccess":[
        "nextStepName"
      ]
    }
  ]
}
```

When you set up a Loop Over Variable step, include the following information:

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y

Parameter	Description	Required
List `list`	A reference to a previous step that provides a list of items to loop through.	Y

Loop Over Variable Step Scope

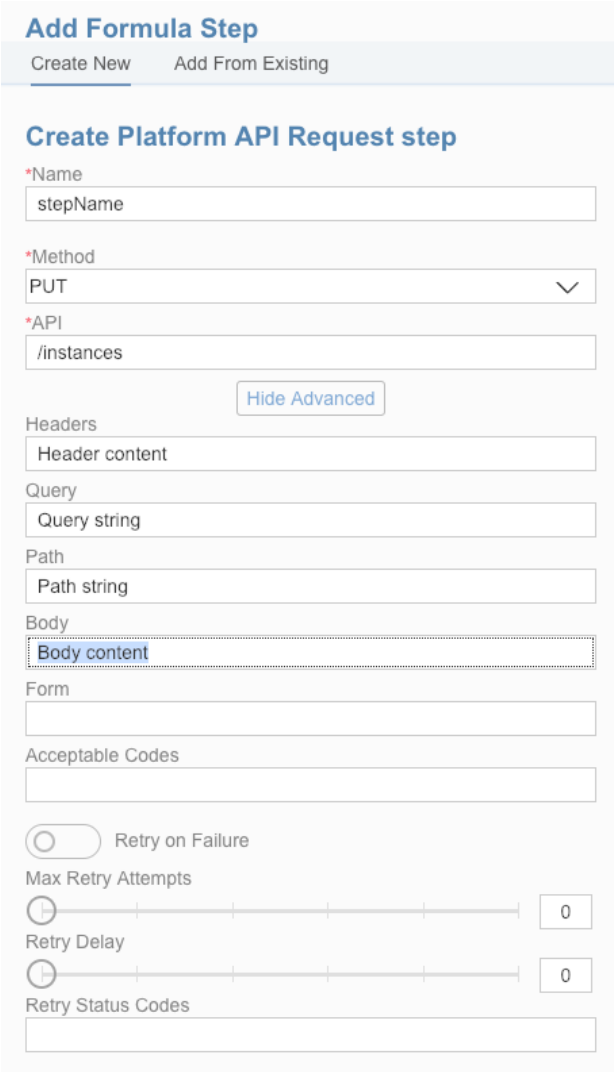
Loop Over Variable steps make available the current object being processed and the index to each step executed inside of that loop. For example, if we have a `loop` step named `looper`, any steps that are run inside of that loop would have access to `looper.index` and `looper.entry`.

Example references to Loop scope:

- `${steps.myLoopStep.entry.id}`
- `${steps.myLoopStep.index}`

Platform API Request

The Platform API Request (`request`) step makes an API call to one of our platform APIs.




```
{
 "steps":[
  {
   "name":"stepName",
   "onFailure":[

   ],
   "type":"request",
   "properties":{
    "elementInstanceId":"${config.elementVariable
}",
    "method":"POST",
    "api":"/instances",
    "headers":"Header content",
    "query":"query string",
    "path":"path string",
    "body":"Body content",
    "acceptableStatusCodes":"200,201",
    "retry":"true",
    "retryAttempts":"5",
    "retryDelay":"401",
    "retryStatusCodes":"500,501"
   },
   "onSuccess":[
    "nextStepName"
   ]
  }
 ]
}
```

When you set up a Platform API Request step, include the following information:

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y
Connector Instance Variable `elementInstanceId`	Specifies the connector instance that receives the API call.	Y
Method `method`	The API method of the API call, such as GET, POST, PUT, PATCH, or DELETE.	Y
API `api`	The endpoint, such as `hubs/crm/contacts`.	Y
Headers `headers`	The headers to pass along as part of the API request. You rarely need to add anything to the headers, but you can use this parameter to pass common header information such as content types.	N
Query `query`	Specifies the filter query to send with the related request. Construct the query in another step and refer to it in the query field. For example, ${steps.previousStep.query}	N
Path `path`	Support earlier formulas where `path` defined variables, such as an {ID} variable in an endpoint. In the latest version, the path parameter is unnecessary.	N
Body `body`	Specifies the JSON body to send with the related request. Construct the JSON body in another step and refer to it in the `body` parameter. For example, `${steps.previousStep.body}`.	N
Acceptable Codes `acceptableStatusCodes`	A comma-separated list (`200,201`) of codes, range (`200-205`), or both (`200-205,208`) returned in the response that indicates success.	N
Retry on Failure `retry`	Indicates that we should retry a configurable number of times if the request fails.	N
Max Retry Attempts `retryAttempts`	The maximum number of times to retry the request.	N
Retry Delay `retryDelay`	The time in milliseconds to wait between retries.	N
Retry Status Codes `retryStatusCodes`	A comma-separated list (`500,502`) of codes, range (`400-415`), or both (`400-415,500,502`) returned in the response that indicates that we should retry the request.	N

Platform API Request Step Scope

Platform API Request steps add the step execution values described in the example JSON below to the formula context. The formula context is then passed from step-to-step, allowing you to use these values in any subsequent steps in your formula.

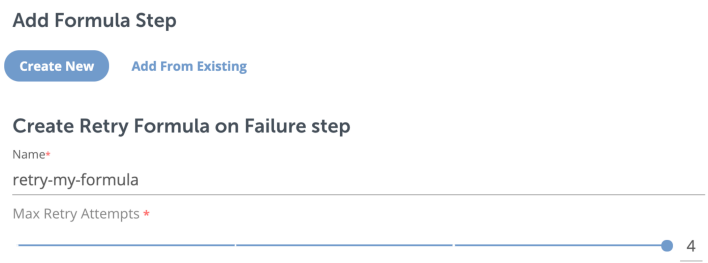
```
{
  "myPlatformStep": {
    "request": {
      "query": "{}",
      "body": "{\"Name\":\"New Account Name\"}",
      "method": "POST",
      "path": "{}",
      "uri": "/elements/api-v2/hubs/crm/accounts",
      "headers": "{\"authorization\":\"Element /ABC=, User DEF=, Organization GHI\",\"content-
length\":\"14\",\"host\":\"jjwyse.ngrok.io\",\"content-type\":\"application/json}"
    },
    "response": {
      "code": "200",
      "headers": "{\"Set-Cookie\":
\"CESESSIONID=2CA15552EE56EAF65BF1102F6CACEACC;Path=/elements/;HttpOnly\"}",
      "body": "{\"Id\": \"001tx3WcAAI\", \"Name\": \"New Account Name\"}"
    }
  }
}
```


Example references to Platform API Request scope:

- `${steps.myPlatformStep.request}`
- `${steps.myPlatformStep.request.body}`
- `${steps.myPlatformStep.response.code}`

Retry Formula on Failure

Retry Formula on Failure (`retryFormulaExecution`) retries a formula instance execution with the same input data. You can configure the number of retry attempts with a maximum of 4 attempts.




```
{
  "steps": [
    {
      "name": "stepName",
      "onFailure": [],
      "type": "retryFormulaExecution",
      "properties": {
        "retryAttempts": "4"
      },
      "onSuccess": [
        "nextStepName"
      ]
    }
  ]
}
```


When you set up a Retry Formula on Failure step, include the following information:

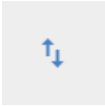
Parameter	Description	Required
Name `name`		Y
Max Retry Attempts `retryAttempts`	The maximum number of times to retry the request.	N

Retry Formula on Failure Step Scope

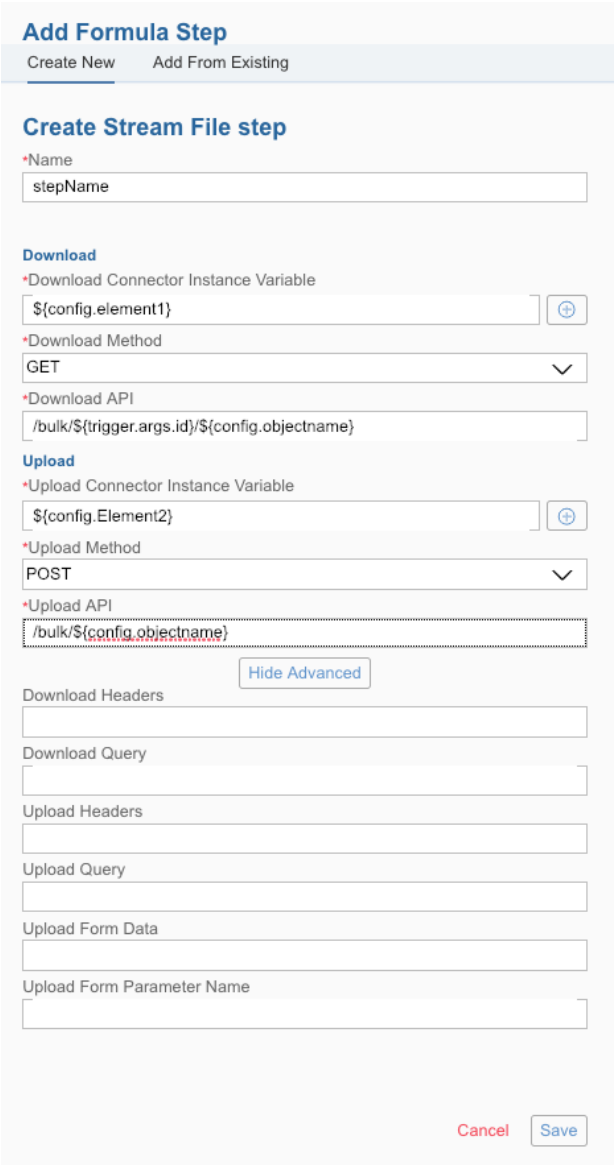
Retry Formula on Failure steps truncate the formula execution and schedule a retry execution for a later time based upon the retry attempt number. The result of this equation is used to schedule a retry in minutes. The step execution response value for this step is a `string` as shown in the example below.

```
{
 "id": "53067",
 "key": "retry.error",
 "value": "formula instance execution scheduled for retry at approximately 2016-12-05T08:52:37-07:00"
}
```

In this example, the step name in the formula is `retry`, and the value of the step execution indicates the time when the formula execution will be retried.

Stream File 

Stream File (`elementRequestStream`) steps move a file from one Connector Instance to another. Stream Files steps configure two API requests instead of just one. One request downloads the date from a connector instance, and the second request uploads the data to another. Use the response body of the download request as the request body of the upload request.




```
{
  "steps": [
    {
      "name": "stepName",
      "onFailure": [],
      "type": "elementRequestStream",
      "properties": {
        "uploadElementInstanceId": "${config.uploadElementVariable}",
        "uploadMethod": "POST",
        "downloadQuery": "Query string",
        "uploadQuery": "Query string",
        "uploadApi": "/bulk/${config.objectname}",
        "uploadHeaders": "${steps.previousStep.uploadHeaders}",
        "uploadFormData": "${steps.previousStep.formData}",
        "downloadMethod": "GET",
        "downloadElementInstanceId": "${config.downloadElementVariable}",
        "downloadHeaders": "${steps.previousStep.downloadHeaders}",
        "uploadFormDataName": "${steps.previousStep.formParameter}",
        "downloadApi": "/bulk/${trigger.args.id}/${config.objectname}"
      },
      "onSuccess": [
        "nextStepName"
      ]
    }
  ]
}
```


To see a Stream File step in action see Bulk Transfer CRM Data.

When you set up a Stream File step, include the following information:

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y
Download/Upload Connector Instance Variable `uploadElementInstanceId` / `downloadElementInstanceId`	Specifies the connector instance that receives the API call.	Y
Download/Upload Method `uploadMethod` / `downloadMethod`	The API method of the API call, such as GET, POST, PUT, PATCH, or DELETE.	Y
Download/Upload API `uploadApi` / `downloadApi`	The endpoint, such as `hubs/crm/contacts` .	Y

Parameter	Description	Required
Download/UploadHeaders `uploadHeaders` / `downloadHeaders`	The headers to pass along as part of the API request. You rarely need to add anything to the headers, but you can use this parameter to pass common header information such as content types.	N
Download/UploadQuery `uploadQuery` / `downloadQuery`	Any query parameters, such as a OCNQL query or pagination, to pass as part of the API request.	N
Upload Form Data `uploadFormData`	Specifies the form data to send with the related request. Construct the form data in another step and refer to it in the Upload Form Data parameter. For example, `${steps.previousStep.formdata}` .	N
Upload Form Parameter Name `uploadFormDataName`	Specifies the name of the form parameter.	N

Stream File Step Scope

Stream File steps add the step execution values described in the example JSON below to the formula context. The formula context is then passed from step-to-step, allowing you to use these values in any subsequent steps in your formula.

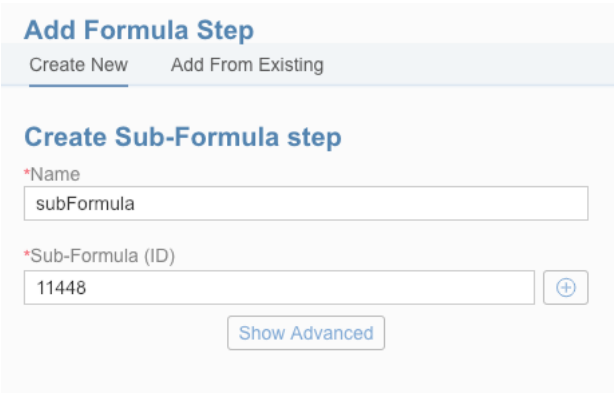

```
{
 "myStreamStep": {
  "download": {
    "request": {
       "query": "{}",
       "method": "POST",
       "uri": "/elements/api-v2/hubs/crm/accounts",
       "headers": "{\"authorization\":\"Element /ABC=, User DEF=, Organization GHI\",\"content-length\":\"14\",\"host\":\"jjwyse.ngrok.io\",\"content-type\":\"application/json}"
      },
      "response": {
       "code": "200",
       "headers": "{\"Set-Cookie\":
\"CESESSIONID=2CA15552EE56EAF65BF1102F6CACEACC;Path=/elements/;HttpOnly\"}"
      }
   },
   "upload": {
     "request": {
       "query": "{}",
       "method": "POST",
       "uri": "/elements/api-v2/hubs/crm/accounts",
       "headers": "{\"authorization\":\"Element /ABC=, User DEF=, Organization GHI\",\"content-length\":\"14\",\"host\":\"jjwyse.ngrok.io\",\"content-type\":\"application/json}"
      },
      "response": {
       "code": "200",
       "headers": "{\"Set-Cookie\":
\"CESESSIONID=2CA15552EE56EAF65BF1102F6CACEACC;Path=/elements/;HttpOnly\"}",
       "body": "{\"Id\": \"001tx3WcAAI\", \"Name\": \"New Account Name\"}"
      }
    }
  }
}
```


Example references to Stream File scope:

- `${steps.myStreamStep.download.request.query}`
- `${steps.myStreamStep.upload.request.headers}`
- `${steps.myStreamStep.upload.response.body}`

Sub-Formula

Sub-formula (`formula`) steps run another formula instance.




```
{
 "steps": [
  {
   "name": "stepName",
   "onFailure": [],
   "type": "formula",
   "properties": {
    "formulaId": "11448"
   },
   "onSuccess": [
    "nextStepName"
   ]
  }
 ]
}
```

When you set up a Sub-Formula step, include the following information:

Parameter	Description	Required
Name `name`	The name of the formula step. The name must be unique within the formula.	Y
Sub-Formula (ID) `formulaId`	The ID of the formula.	Y
`args`	Any values that should be made available to the sub formula.	N
`subFormulaConfigs`	Any variables required for the sub formula.	N

Sub-Formula Step Scope

Sub-formula steps add the values produced as the result of the last step in the sub-formula. Therefore, we recommend that when you build formulas to be used by other other formulas that you add a specific step to aggregate and returns whatever data is needed in the parent's formula context.

If the sub-formula requires variables, then those variables can either be set in the parent formula instance using the same config names or passed in via the `subFormulaConfigs` property. All sub-formulas inherit their parent formula's configuration values. If you pass in the `subFormulaConfigs` these are added to the list of existing configs from the parent and the sub-formula has access to the parent's configs and those passed in with the values in `subFormulaConfigs` taking precedence.

The `args` can be accessed in the sub-formula using `trigger.args` . The `subFormulaConfigs` can be accessed in the

sub-formula using config for example: `${config.crmInstanceId}` .