Source URL: [NITRO™ Workflows > Configuring Workflows > Conditions](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?conditions.html)

**Conditions**

Conditions allow you to restrict the execution of your workflow so that it only runs when specified data conditions are met. Conditions support the following value types for a list column:

•Literal/Custom values

•Another field value

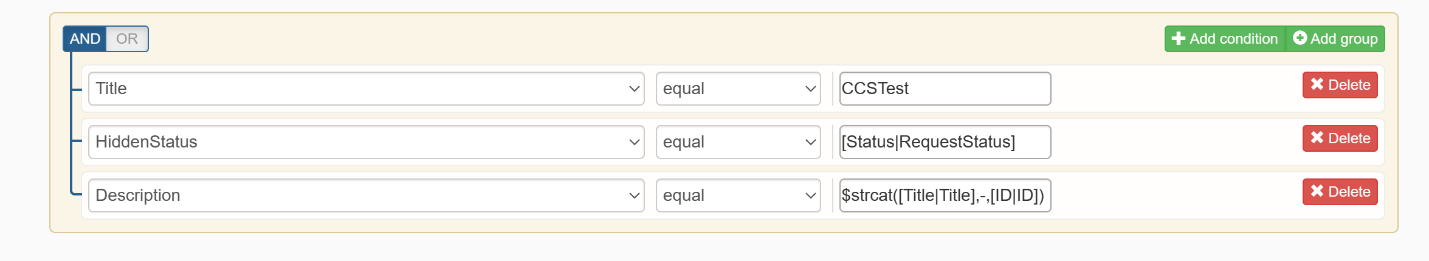
•[Functions](https://www.crowcanyon.info/nitro/appmanual_v2/functions.html)

•[Place Holders](https://www.crowcanyon.info/nitro/appmanual_v2/placeholders.html)

**1.Single Line of Text**

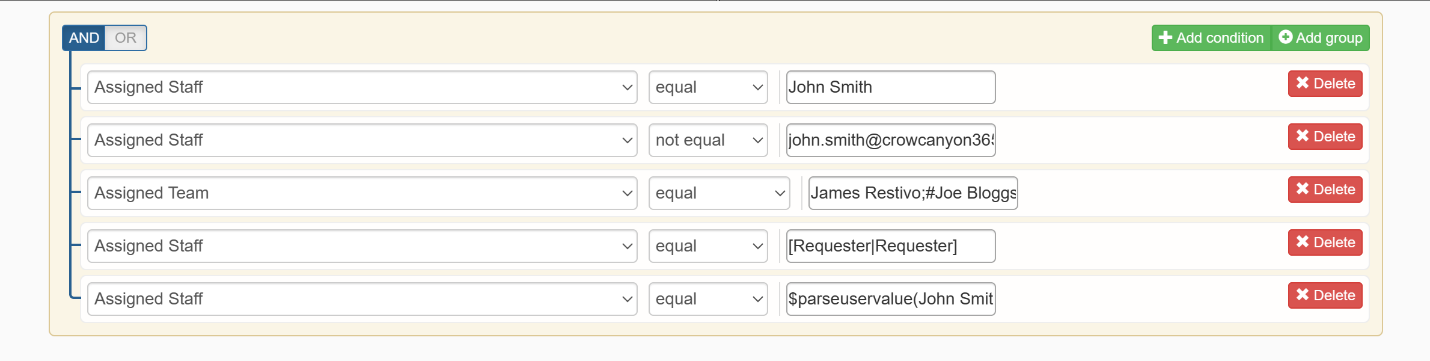
|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Column Value - Fixed value | CCSTest |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Status|RequestStatus] |
| Function | $Strcat  $formatvalue |
| Query List Value - QueryListName##[ColumnDisplayName|ColumnInternalName] | GetProblem##[Title|Title]. **Note**: Can not use this in workflow condition. you can use this in gateway condition. |

Sample example:



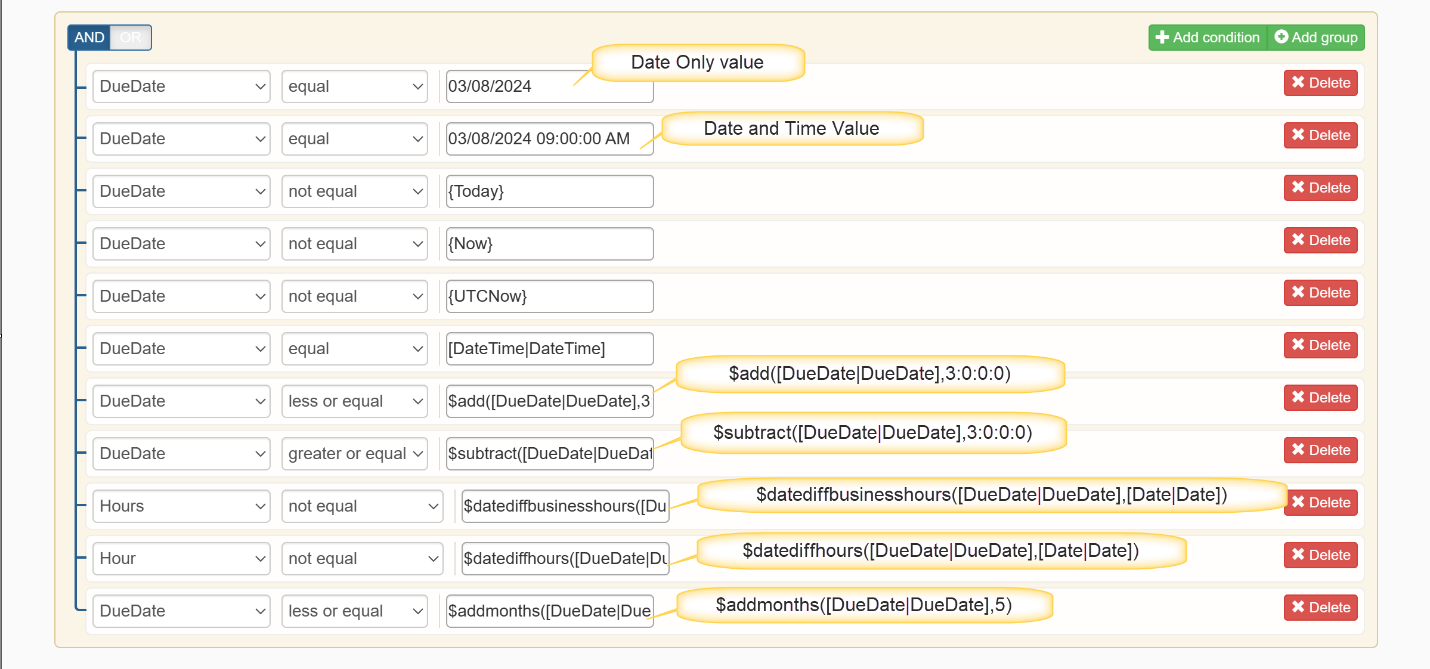
**2.Person or Group**

|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Column Value | - Pavan Kumar - [pavan@crowcanyon365.com](https://www.crowcanyon.info/nitro/appmanual_v2/conditions.html) - pavan;#james |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Requester|Requester] |
| Function | - parseuservalue  Ex- $parseuservalue(John Smith)  - usercollection |
| Query List Value - QueryListName##[ColumnDisplayName|ColumnInternalName] | GetConfiguration##[Approver|Approver] |
|  |  |



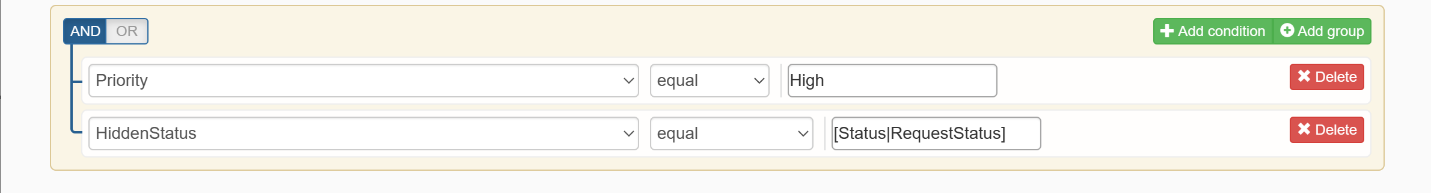
**3.Date Time**

|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Column Value | 01/01/2018 |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Due Date|DueDate] |
| Function | - subtract  - add  - calcbusinesstime  - addmonths  - datediffhours  - datediffbusinesshours |
| Query List Value - QueryListName##[ColumnDisplayName|ColumnInternalName] | AssetItem##[ExpiryDate|ExpiryDate] |
| Placeholder | {Today} |



**4.Choice**

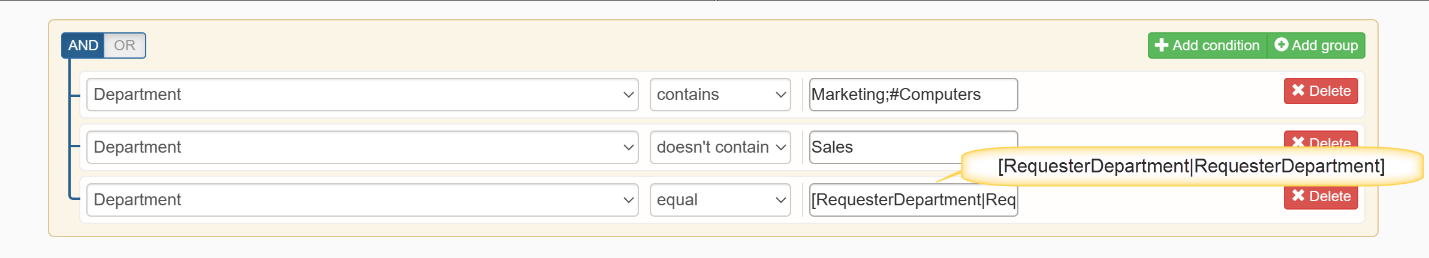
|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Column Value | High |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Priority|Priority] |
| Function | Strcat  xpath |
| Query List Value - QueryListName##[ColumnDisplayName|ColumnInternalName] | TicketItem##[Priority|Priority] |



**5.Multi Choice**

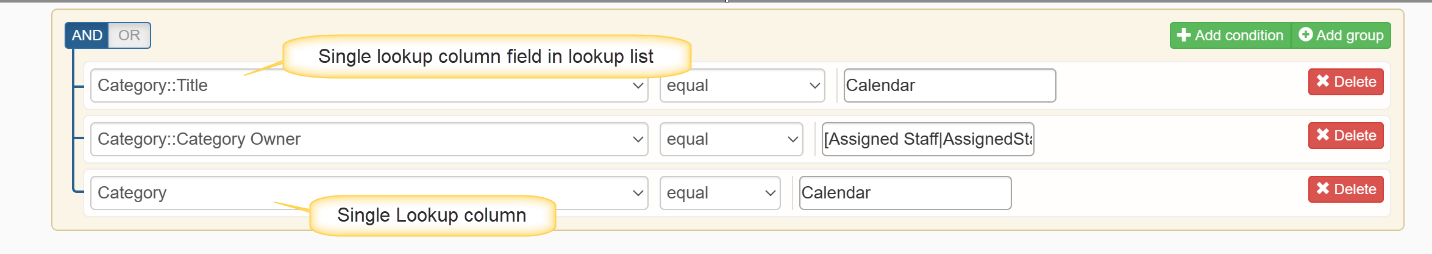
|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Column Value | Marketing;#Computers |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Priority|Priority] |
| Function | Strcat |
| Query List Value | TicketItem##[Priority|Priority] |

**NOTE:** It is not necessary that multiple choice values specified in the item field and condition value be in same order. As long as both contain the same set of choices, condition will evaluate to true. For instance in the condition it can be either Choice1;#Choice2 or Choice2;#Choice1.



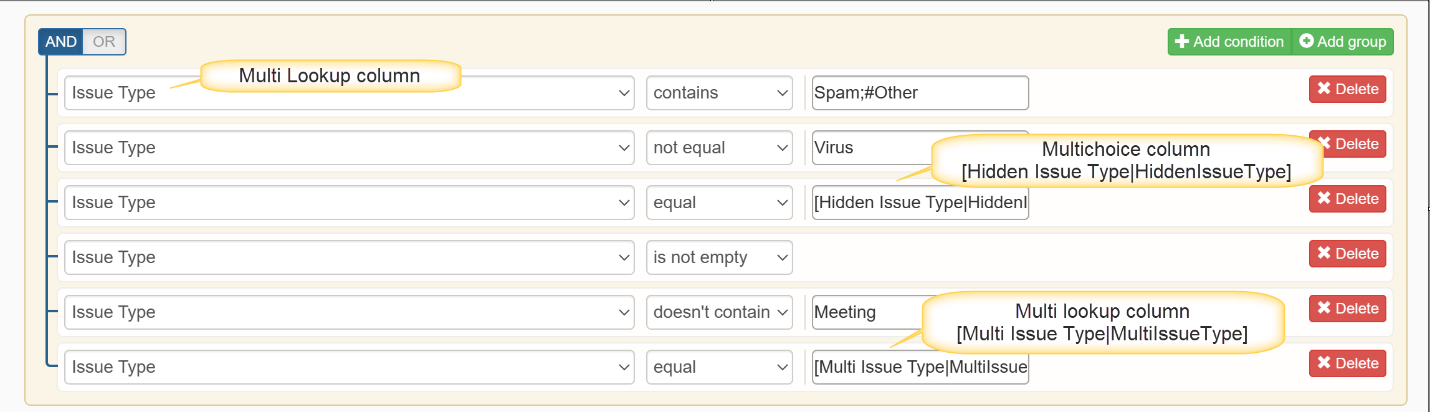
**6.Lookup**

|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Assigned Staff|AssignedStaff] |
| Column Value | Calendar |
| Query List Value - [ColumnDisplayName|ColumnInternalName] | TicketItem##[Assigned Staff|Assigned staff] |



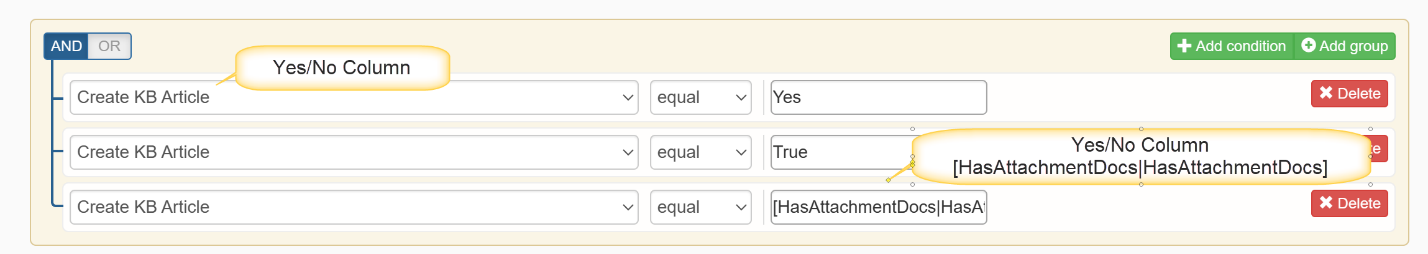
**7.Multi Lookup**

|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Text|Text] |
| Column Value | Spam;#Calendar |
| Query List Value | TicketItem##[Text|Text] |



**8.Yes/No**

|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [HasAttachmentDocs|HasAttachmentDocs] |
| Column Value | - True - False - Yes  - No |
| Query List Value | TicketItem##[SendForApproval|SendForApproval] |



**9.Number or Currency**

|  |  |
| --- | --- |
| **Possible Values** | **Example** |
| Another Column name – [ColumnDisplayName|ColumnInternalName] | [Cost|Cost] |
| Column Value | 100 |
| Query List Value | PurchaseItem##[Price|Price] |

Source URL: [NITRO™ Workflows > Configuring Workflows > Placeholders](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?placeholders.html)

**PlaceHolders**

**Note:**The following placeholders should not be used in column names, as they will conflict with workflows:

▪Parentheses “()”

▪Number sign “#”

▪Percent “%”

▪Double Colon "::"

▪Dollar sign “$”

▪At sign “@”

▪Brackets “[]”

▪Braces “{}”

▪Vertical bar “|”

-**afterchange**: used in conditions to check if the value of a column has been changed as part of the current update process.

**Syntax: afterchange:value**

**Possible values:**

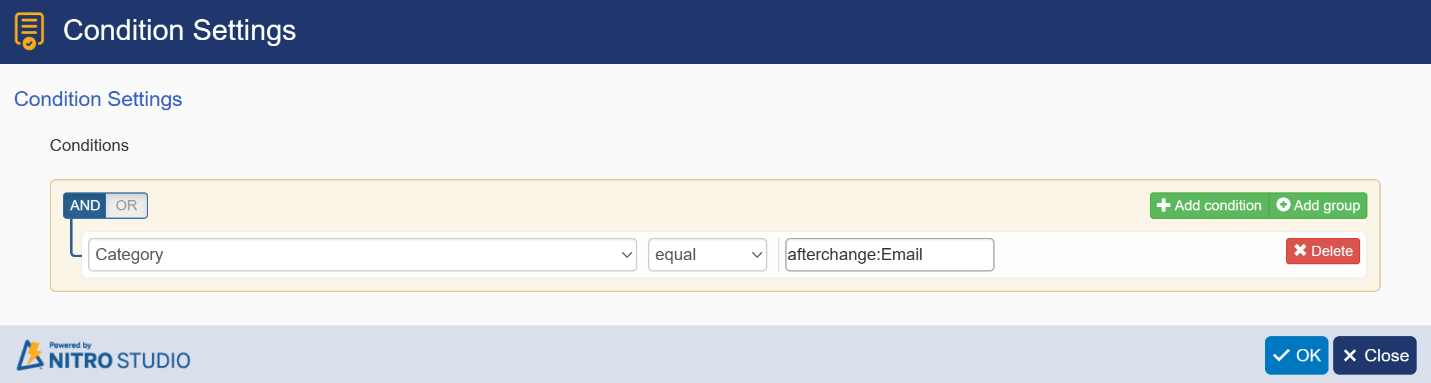
1. afterchange:Fixed value

2. afterchange:[Column Display Name|Column Internal Name]

3. afterchange:[[Any]]

4. afterchange:[[None]]

Example: afterchange:Email



-**beforechange**: used in conditions to check the value of a column before it was updated to another value by the current update process.

**Syntax: beforechange:value**

**Possible Values:**

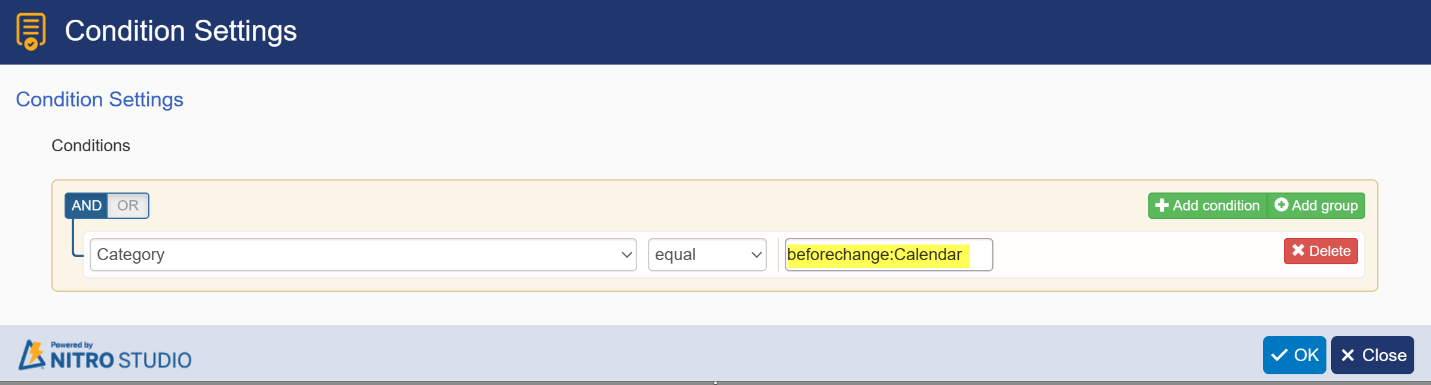
1. beforechange:Fixed value

2. beforechange:[Column Display Name|Column Internal Name]

3. beforechange:[[Any]]

4. beforechange:[[None]]

Example: beforechange:Calendar



**Note:**

These placeholders should only be applied on workflows that run-on Item modified event.

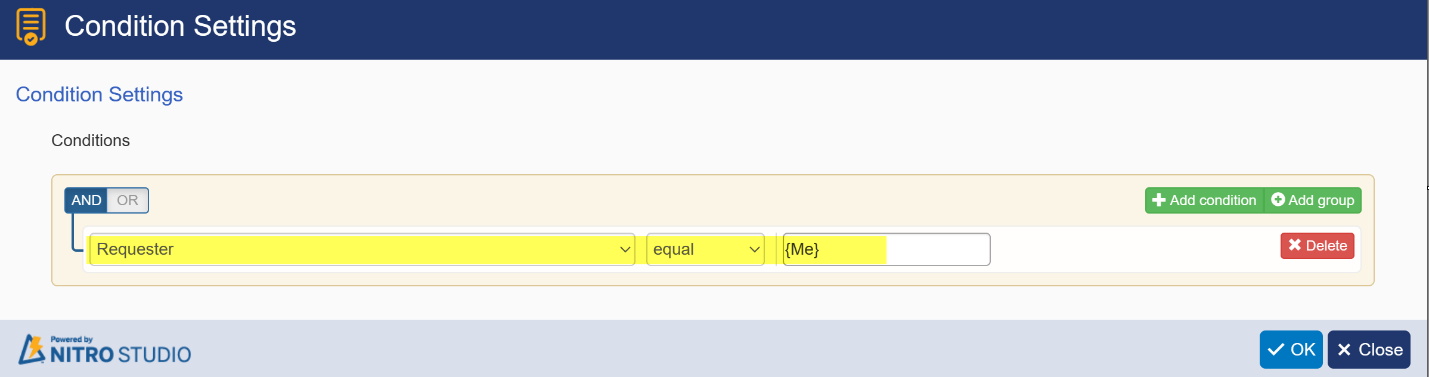
Use **afterchange:[[Any]]** for any change in a column value.

Use **afterchange:[[None]]** for no change in column value.

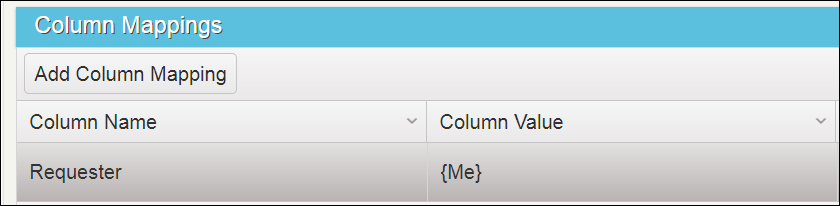
-**Me:**Used to check a Person/Group column value against the current logged in user.

-**Note:**This option is available only in on-premises only.

In Conditions:

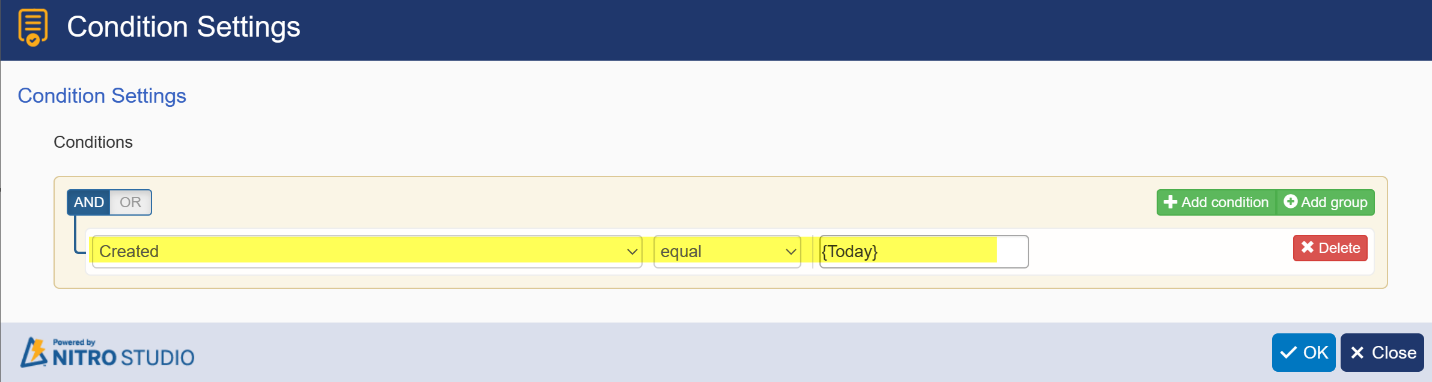


*In Field Mappings:*

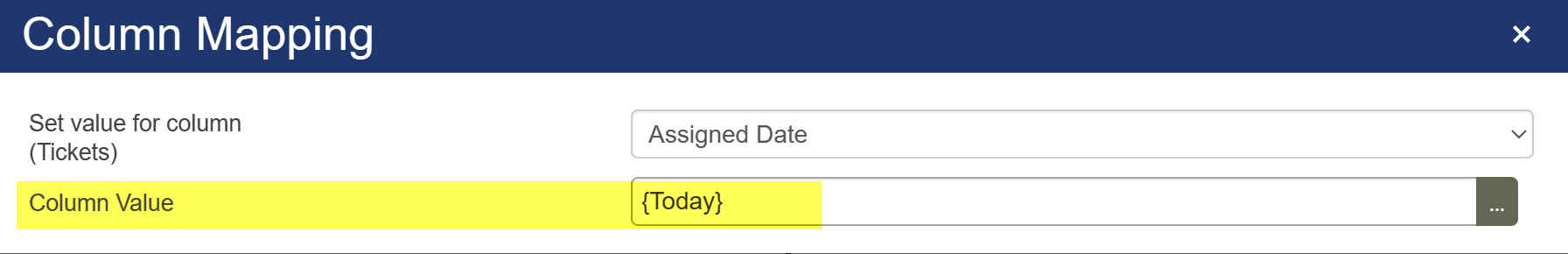


-**Today:**Used to check Date Time column value with today’s date

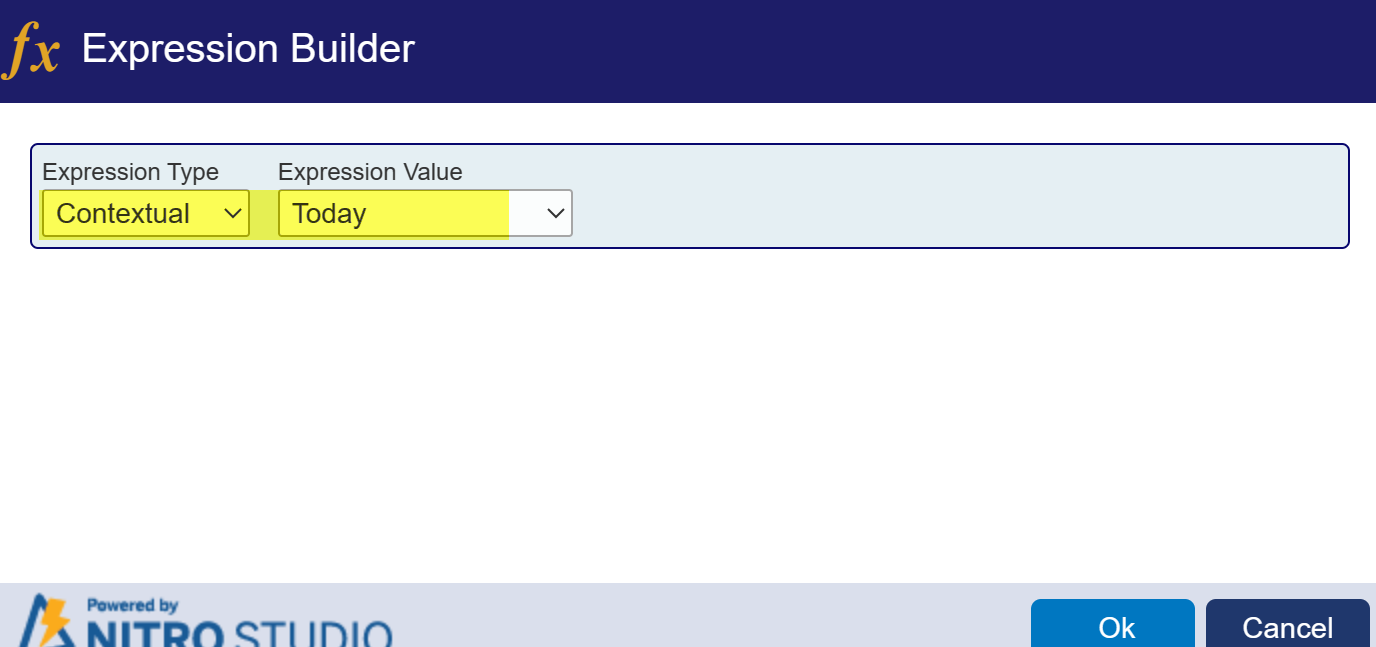
*In Conditions:*



*In Field Mappings:*

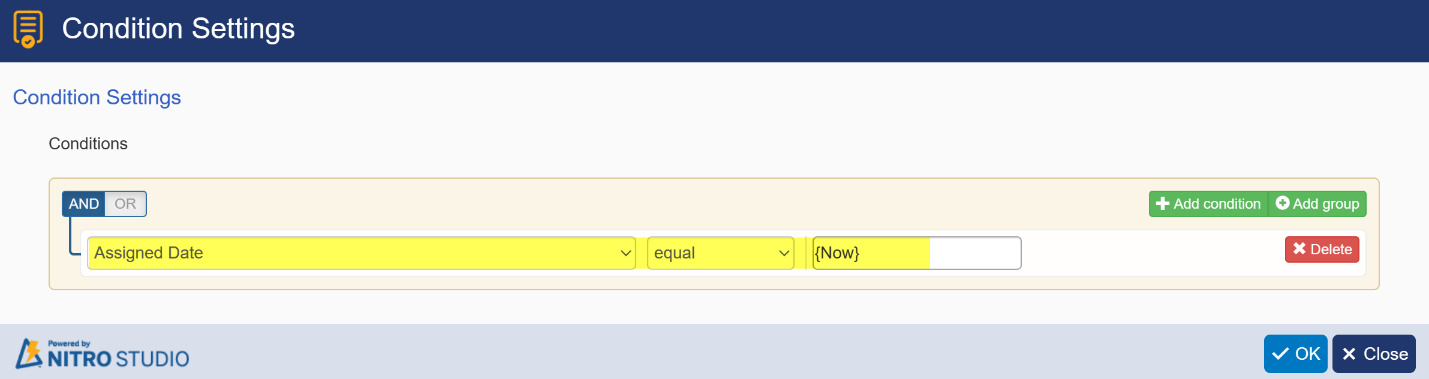


*In Field Mapping with expression builder:*

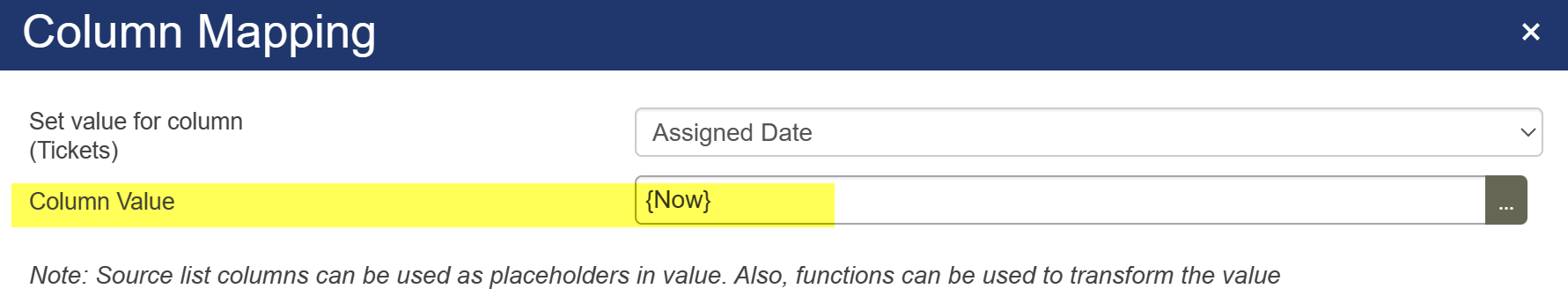


**Now:**Gives current date time. Used to Update Date Time value with current time.

*In Conditions:*

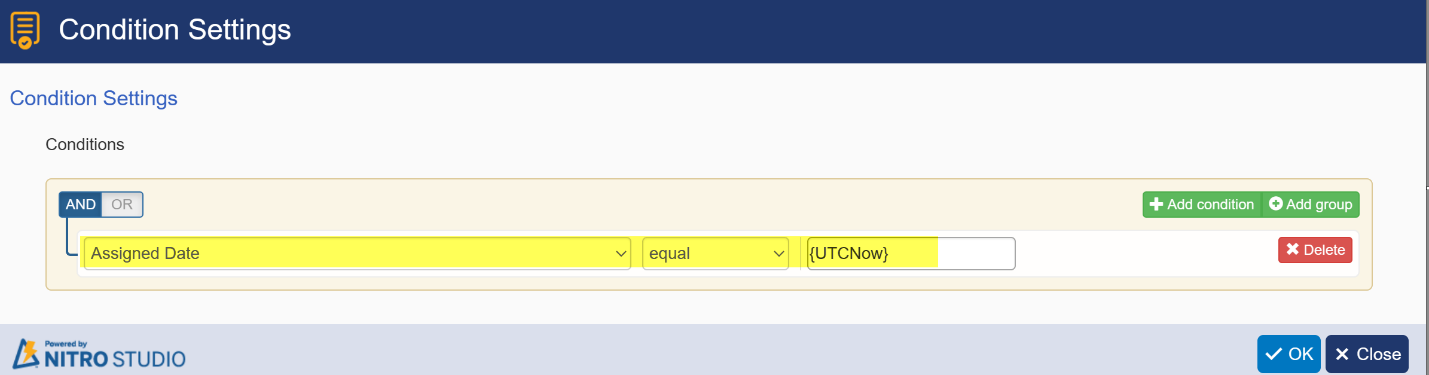


*In Field Mappings:*

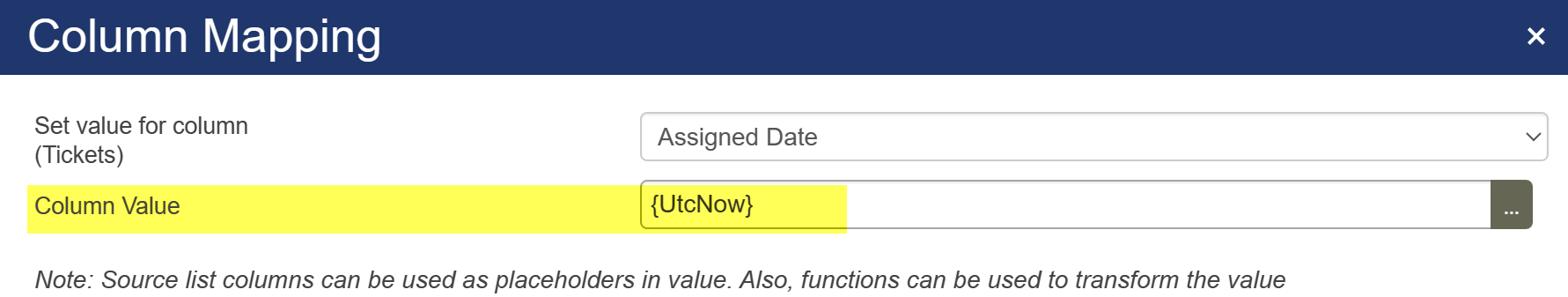


**UtcNow:**Gives current date time in UTC format. Used to update Date Time value with current UTC time.

*In Conditions:*



*In Field Mappings:*

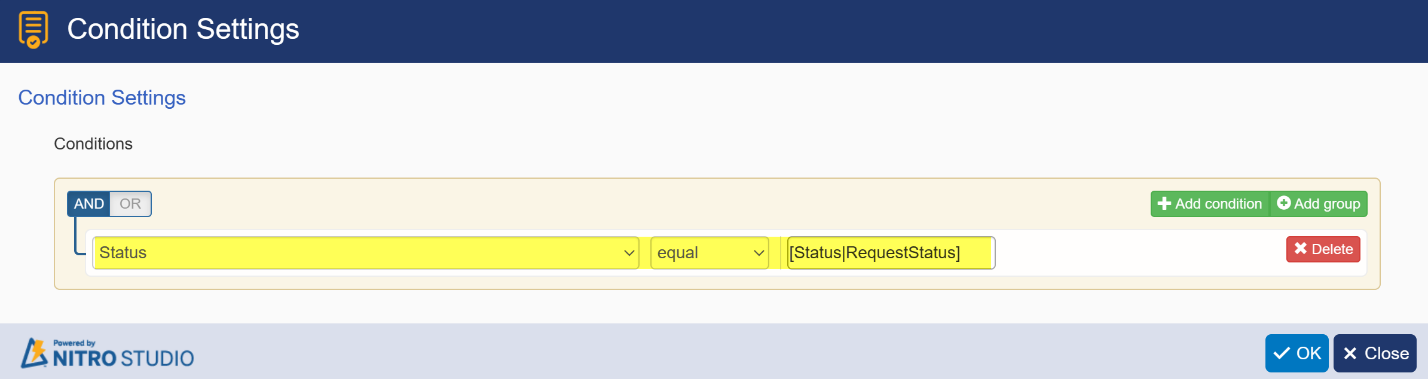


**Column Placeholder: in condition,** it is usedto compare value between two columns with different type of operators. **In field mapping**, it is used to update the column value with another column value.

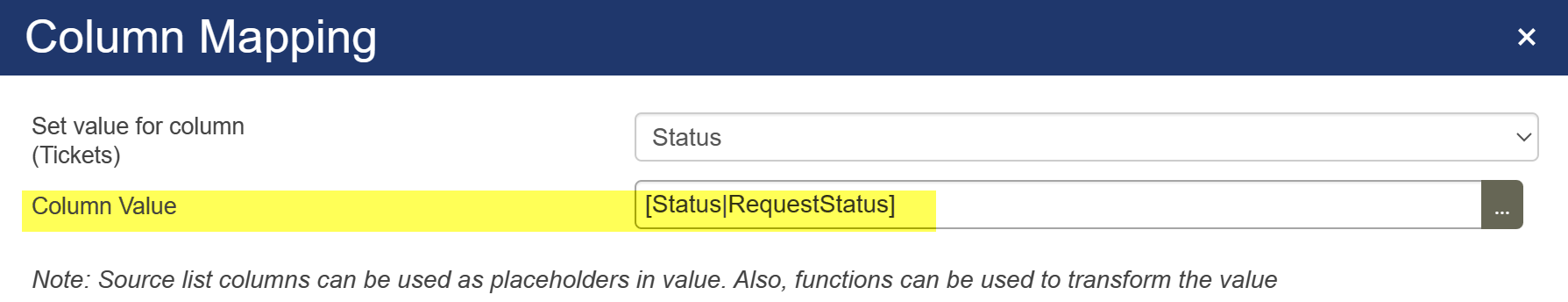
**Syntax: [Column Display Name|Column Internal Name]**

Ex: [Status|RequestStatus]

*In Conditions:*



*In Field Mappings:*

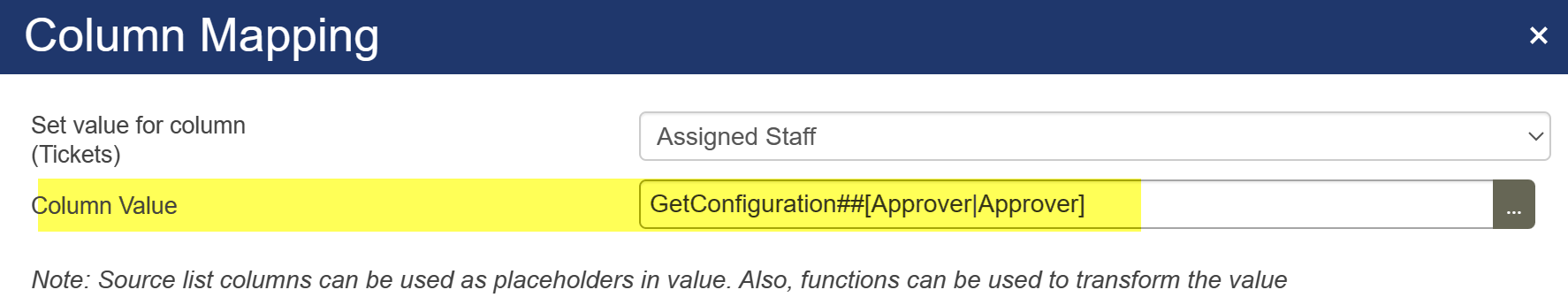


**Query list item placeholder:**it is used to update the column value with query list item column value. Note: this can not be used in workflow condition, because query list action will be executed after workflow condition evaluation. you can use this in Gateway condition and field mapping.

**Syntax: Query list action name##[Column Display Name|Column Internal Name]**

Ex: GetConfiguration##[Approver|Approver]

*In Field Mappings:*

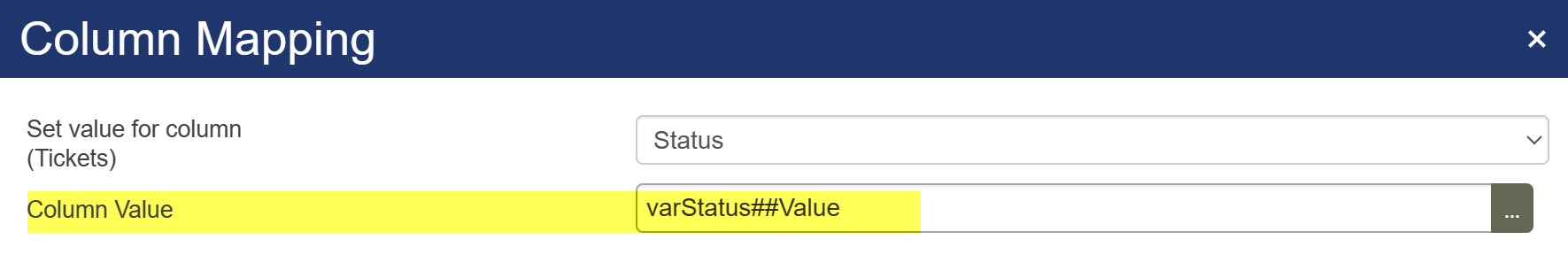


**Variable:**The variable value can be updated at run time using query list actions and create item actions and this updated variable values can be used in conditions, column mappings in further actions. Note: this can not be used in workflow condition, because variable value will be evaluated after workflow condition evaluation. you can use this in Gateway condition and field mapping.

**Syntax: Variable##Value**

Ex: VarStatus##Value

*In Field Mappings:*



Source URL: [NITRO™ Workflows > Configuring Workflows > Reserved Characters](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?reserved-characters.html)

Reserved characters in Crow Canyon Workflow Manager:

There are few special "reserved" characters that are used in Crow Canyon's NITRO Workflow Manager and they cannot be used in conditions and field mappings. If these characters are used in field mappings or in conditions, the workflow actions won’t be executed since it will treat them as placeholders.

List of reserved characters in our Workflow Manager:

Parentheses “()”

Number sign “#”

Percent “%”

Double Colon "::"

Dollar sign “$”

At sign “@”

Brackets “[]”

Braces “{}”

Vertical bar “|”

From URL: [Custom Actions > Conditions > Advanced Condition](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?advanced-condition.html)

**Advanced Condition** shows up in the list of Columns when in Condition Settings even though it is not actually a Column. It is there for the convenience of using this feature while in Conditions.

**Advanced Condition** is used to check the conditions of current/other lists column values using the following. Also, check the Use Cases that are below the image.

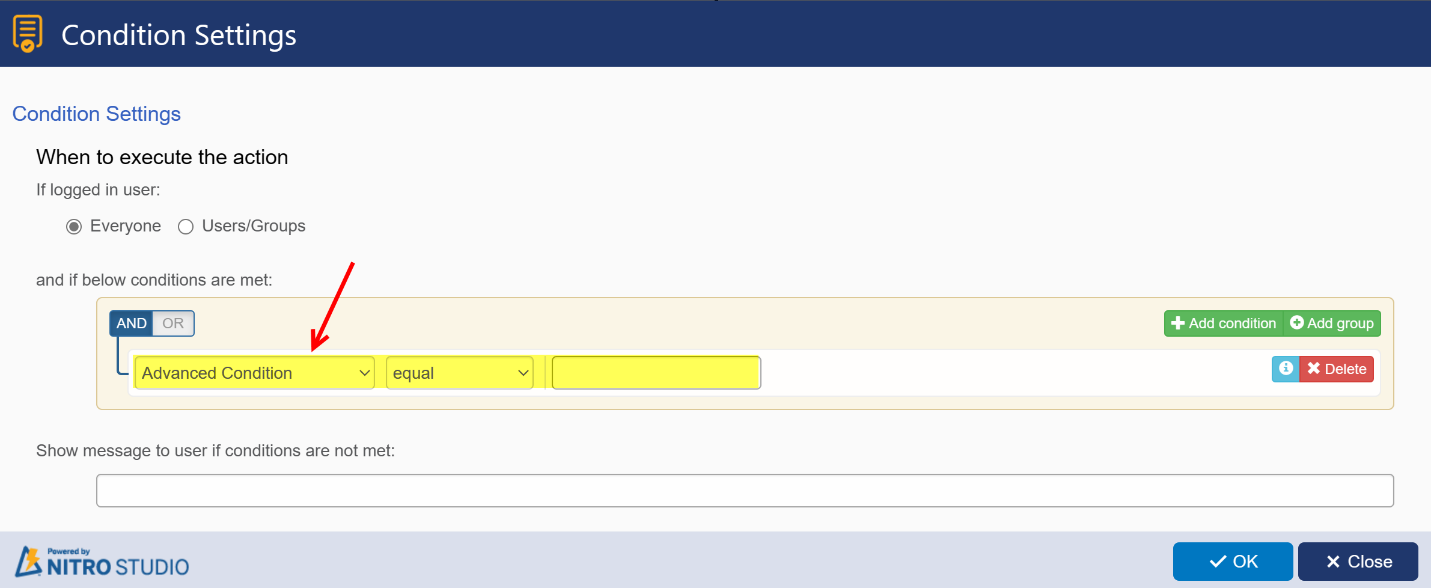
•Literal/Custom values

•Place Holders

•Query List

•Variables

Separator for placeholders: ,, (double comma)



**Use Case 1:**

Suppose configuration item is having “Send for Approval” if custom action has to be executed based on “Send for Approval” is true then get the configuration item using query list action and in next action use Advanced Condition is equal to Config##[Send for Approval|SendforApproval],,true

**Use Case 2:**

Suppose Purchase request is having 4 assets if the assets total cost is greater than 1000 then send for approval. To check this condition using query list action get the 4 products linked to the purchase requests and in next custom action define Advanced Condition is greater than PRAssets##$sum([Cost|Cost]),,1000

**Syntax:**

**1.Placeholders:**

•[field placeholder1],, [field placeholder2] – For single text field

•Literal/Custom value,, [field placeholder] – For single text field

•$parsefieldvalue([field placeholder],Value),,$parsefieldvalue([field placeholder],Value)

**2.Variables:**

•[field placeholder],,variable##Value – For single text field

•Literal/Custom value,,variable##Value – For single text field

•$parsefieldvalue(variable##Value,Value),,$parsefieldvalue(variable##Value,Value)

•Literal/Custom value,,$parsefieldvalue(varNumber##Value,Value)

•Literal/Custom value; Literal/Custom value,,$parsefieldvalue(variable##Value,Value)

**3.Query List:**

•[field placeholder],,querylist## placeholder – For single text field

•Literal/Custom value,, querylist ##placeholder – For single text field

•$parsefieldvalue(querylist ##placeholder,Value),,$parsefieldvalue(querylist ##placeholder,Value)

•Literal/Custom value,,$parsefieldvalue(querylist ##placeholder,Value) – For single selection field

•Literal/Custom value; Literal/Custom value,,$parsefieldvalue(querylist## placeholder,Value) – For multi selection field

**Number/Currency Field:**

•$parsefieldvalue(placeholder,Value),,$parsefieldvalue(variable##Value,Value)

•Literal/Custom value,,$parsefieldvalue(variable##Value,Value)

•Literal/Custom value,,$parsefieldvalue(querylist##placeholder,Value)

**Date & Time Field:**

•$parsefieldvalue(placeholder,Value),,$parsefieldvalue(variable##Value,Value)

•Literal/Custom value,,$parsefieldvalue(variable##Value,Value)

•Literal/Custom value,,$parsefieldvalue(querylist##placeholder,Value)

**Choice Field:**

**For Choice field (Single selection)**

•[field placeholder],,variable##Value

**For Choice field (Multiple selection):**

•$parsefieldvalue(variable##Value,Value),,$parsefieldvalue(variable##Value,Value)

•$parsefieldvalue(querylist##placeholder,Value),,$parsefieldvalue(querylist##placeholder,Value)

•$parsefieldvalue(variable##Value,Value),, Literal/Custom value;Literal/Custom value

**Lookup Field:**

**For lookup field (Single selection):**

•[ID],, $parsefieldvalue(variable##Value,ID)

•Literal/Custom value,, $parsefieldvalue(variable##Value,Value)

•Literal/Custom value,, $parsefieldvalue(querylist##placeholder,Value)

**For lookup field (Multiple selection):**

•$parsefieldvalue(variable##Value,Value),,$parsefieldvalue(variable ##Value,Value)

•$parsefieldvalue(placeholder,Value),,$parsefieldvalue(variable##Value,Value)

•Literal/Custom value; Literal/Custom value,,$parsefieldvalue(variable ##Value,Value)

•Literal/Custom value; Literal/Custom value,,$parsefieldvalue(querylist ##placeholder,Value)

**Person or Group Field:**

**For Person or Group field (Single selection):**

•Literal/Custom value,, $parsefieldvalue(variable##Value,Value)

•$parsefieldvalue(placeholder,Value),,$parsefieldvalue(variable##Value,Value)

•$parsefieldvalue(placeholder,Value),,$parsefieldvalue(querylist##placeholder,Value)

**For Person or Group field (Multiple selection):**

•$parsefieldvalue(variable##Value,Value),,$parsefieldvalue(variable ##Value,Value)

•Literal/Custom value;Literal/Custom value,,$parsefieldvalue(variable ##Value,Value)

•Literal/Custom value;Literal/Custom value,,$parsefieldvalue(querylist ##placeholder,Value)

**Note:**

•We can use these conditions to check multi choice, multi lookup and multi person/group types, but we need to have those values in same order (Ex: [A, B = A, B] but not [A, B = B, A])

•Other than single or text column type, need to use ParseFieldValue function (Ref  “Custom Actions\_Variables.docx” document).

•Must keep the Advanced Condition in left side with operator type in middle and condition to check in right text field

Source URL: [Custom Actions > Variables > How to use variables in actions](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?how_to_use.html)

|  |  |
| --- | --- |
| How to use variables in actions **Use in “column mappings” if variable value is simple text or object** |  |

a.Syntax: VariableName##Value

b.Example: varUsers##Value

|  |  |
| --- | --- |
| **Use in “column mappings” if variable value is an object that has different keys** |  |

a.Syntax: VariableName##KeyName

b.Example 1: my variable name is varUsers and it has user value object then to read the user Id, use “varUsers##Id”

c.Example 2: my variable name is varUsers and it has user value object then to read the user email, use “varUsers##Email”

d.Example 3: my variable name is varUsers and it has lookup column value object then to read lookup value, use “varUsers##Value”

e.Example 4: my variable name is varUsers and it has the result of $xpath() function like “{id: 1, City: Hyd}”, then to read city value, use “varUsers##City”

* Supported keys are same as the keys supported in $parsefieldvalue() function.

|  |  |
| --- | --- |
| **Use in “CAML Query” in Query list action** |  |

Syntax is same as described in above column mappings however we need to place this variable syntax inside “%%”. For example, “%%varUsers##Value %%”

|  |  |
| --- | --- |
| **Use in “Run for” in Create item action if variable value is objects collection or array of objects** |  |

When we run “Add List Item” action for a variable, we have two different possibilities to use variable values in column mappings in that “Add List Item” action.

**1.**We can set full object into one column mapping using **@{CollectionValue}**

**2.**If variable object is having any key value pairs then we can set value in column mapping using **@{CollectionValue}KeyName**

Source: [Custom Actions > Variables > How to set dynamic values for variables](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?howtosetdynamicvaluesforvariables.html)

## How to set dynamic values for variables

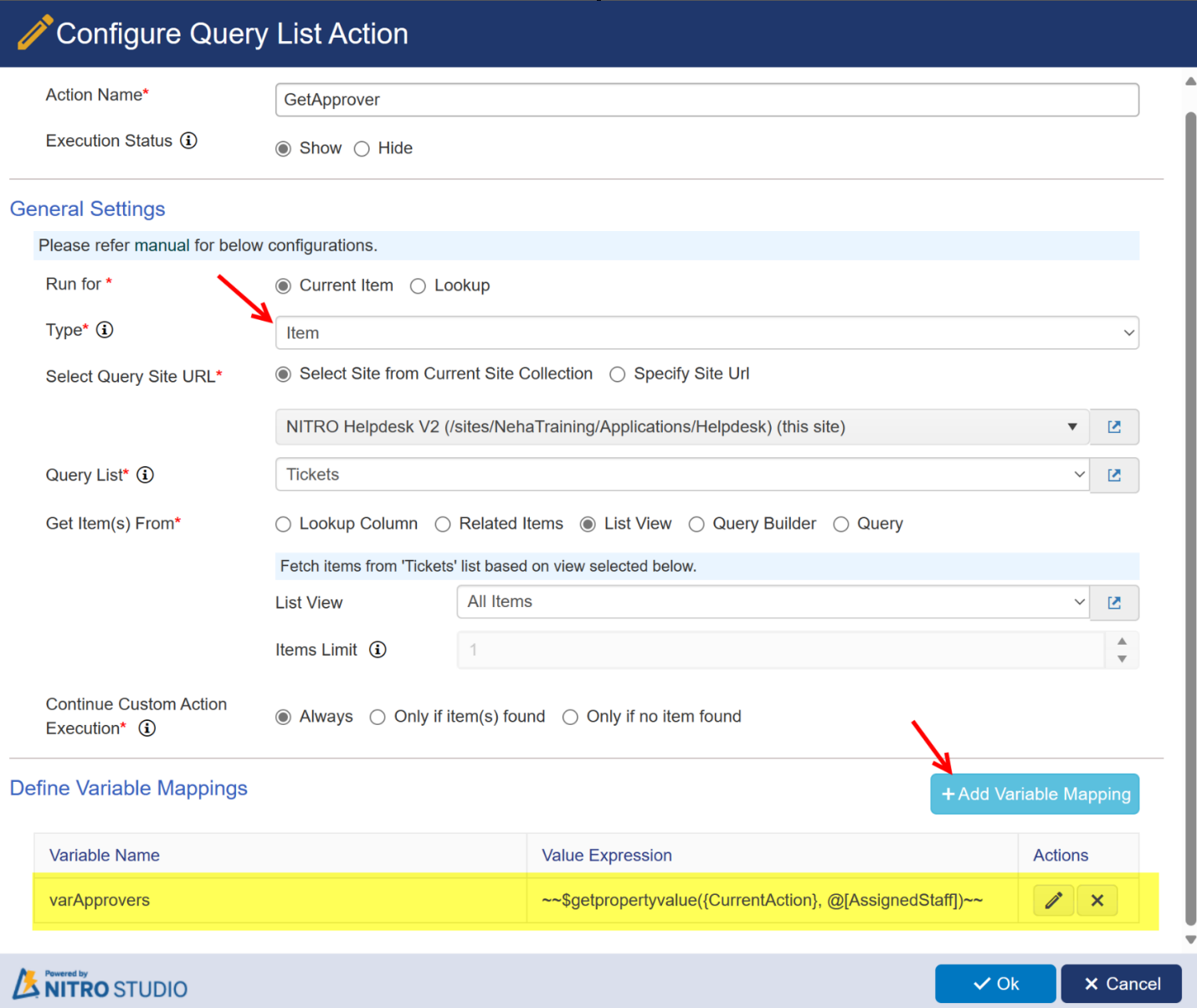
|  |  |
| --- | --- |
| **Query List** |  |

Query list action could return single or collection of items based on the “Type” selection (see screenshot below). If the query list action is configured to return single item, then we can read that return item column values and store in variables for later usage.

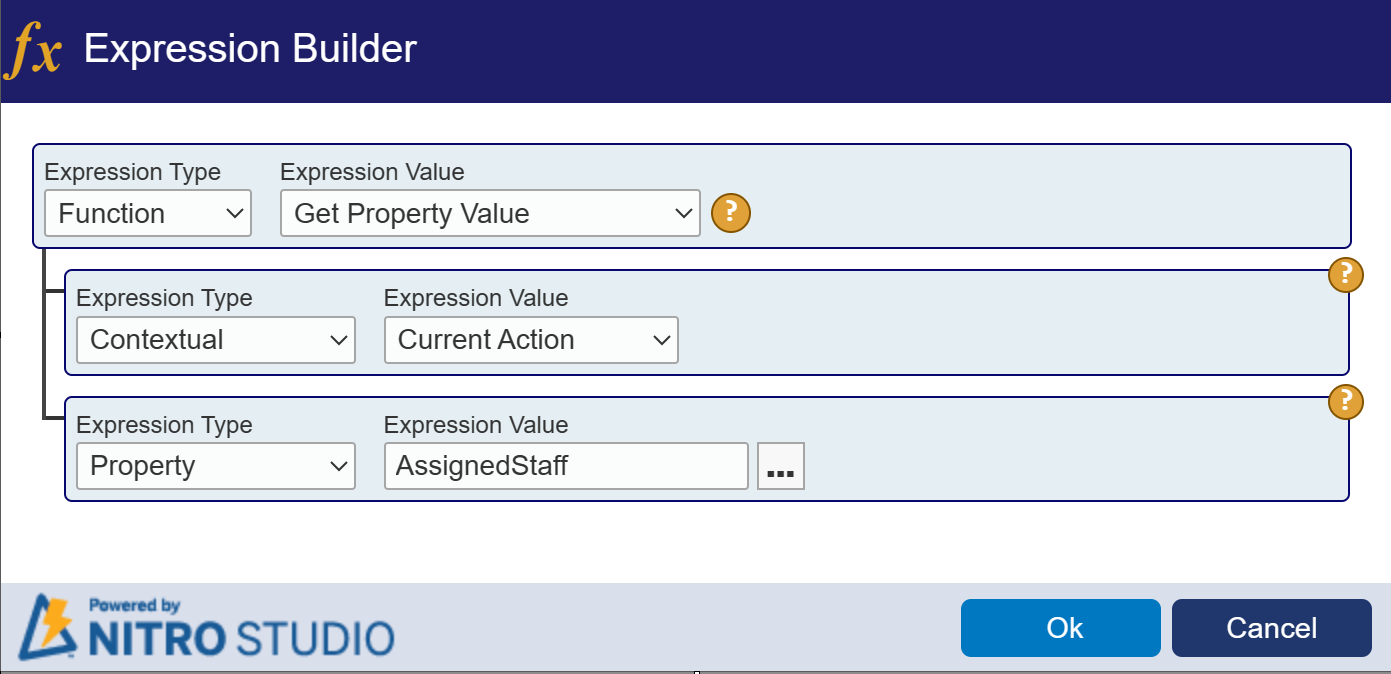
Variable value Supports following types:

* •Literal/Custom values
* •Another field values (Place holders of current item and query list returned item are supported)
* •Functions

Using the “Define Variable Mappings” section in the Query List action settings page, we can assign values to variables (See screenshots below).



**With expression builder:**



**Without expression builder:**



**Syntax for defining variable with query list placeholders within Query List Action: {CurrentAction}##[Assigned Staff|AssignedStaff]**

**Syntax for Query List action result value if used within another action column mapping or variable mapping: Query List action name##[Column display name|Column Internal Name]**

If we store the return item’s column value like in above screenshot, it stores the object value in the variable.

For example, if we store the multiple selections enabled person or group column value, then it stores the collection of selected users so later we can use this variable either in column mappings or use this to run and create one item per each user in this variable value. This is same if we store the multiple selection enabled lookup column value.

If we store literal/custom value, simple column place holders like text column, choice column etc. that returns simple string values, we can use these variables only in the column mappings and we cannot use them to run create item actions.

Refer “[How to use variables in actions](https://www.crowcanyon.info/nitro/appmanual_v2/how_to_use.html)” section for more details on how to read the required value from variable to use in column mappings.

|  |  |
| --- | --- |
| **Invoke Web Service** |  |

Web service call could return simple text or xml text or JSON text as output. We can store either of these output values as text values in variables or we can use functions like $xpath() or $jsonpath() functions to extract the required values and store them in the variables for later use.

If we use $xpath() function that returns array of objects then we can use the variable that stores this function return value and run for this variable to create multiple items. If the function returns single text value for the given xpath expression, then we can use this variable only in column mappings.

Example Web Service Call Result:

<Cities>

 <City id="1">Hyd</City>

 <City id="2">Bng</City>

 <City id="3">Delhi</City>

</Cities>

Sample Usage of Result:

**Expression 1**: $xpath(WebServiceActionName##Value,/Cities)

Output (Array of Objects):

Output1: [Cities: "<City id="1">Hyd</City><City id="2">Bng</City><City id="3">Delhi</City>"]

Output2: <City id="1">Hyd</City>

 <City id="2">Bng</City>

 <City id="3">Delhi</City>

Please note that $xpath() function internally contains two different output values as showed in above example. In this example, Output 1 is JSON object and it will be used in “Run for” in Create item Action. Output 2 is simple text and will be used in column mappings.

**Expression 2:** $xpath(WebServiceActionName##Value,/Cities/City[1])

**Output (Simple Text):**

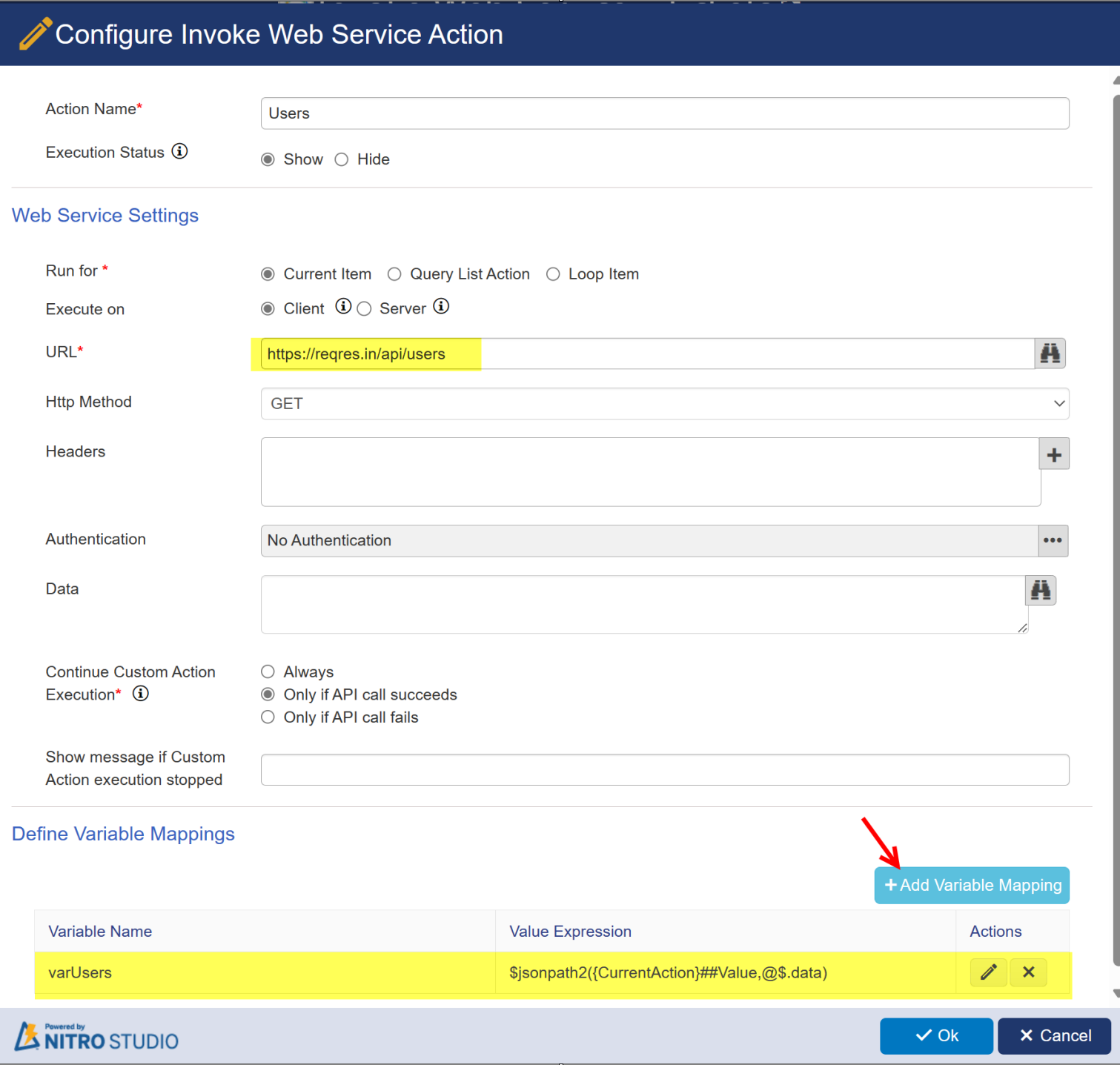
Output1: {id: 1, City: Hyd}

Output2: Hyd

Similarly, if we use $jsonpath() function that returns multiple objects then we can use the variable that stores this function return value and run for this variable to create multiple items. If the function returns simple text then we can use this only in column mappings.

Refer $xpath() and $jsonpath() functions for more details.

Using the “Define Variable Mappings” section in the Invoke Web Service action settings page, we can assign current action values to variables using syntax defined below (See screenshots below).



Variable value Supports following types

•Literal/Custom values

•Another field values (Place holders of current item and query list item actions that are executed before this web service action are supported)

•Functions

**Syntax for defining variable with Invoke Web Service action placeholders within same action:**{CurrentAction}##Value

**In Add/Update action you can run for 'varUsers' variable and update column values using syntax:**

@{CollectionValue}first\_name

@{CollectionVaue}Id

**Syntax for invoke web service action result value if used in another action column mapping or variable mapping**:**{Invoke Web Service Action name}##Value**

**Example: Users##Value**

|  |  |
| --- | --- |
| **Execute Script** |  |

We can write any custom script as part of the execute script action, we can set constant value or any function return value into variable using the predefined method “ccs\_setwfvariablevalue” (See sample screenshot below).

Syntax:

**ccs\_setwfvariablevalue**("variable name", "value");