Source URL: [Custom Actions > Variables > Functions](https://www.crowcanyon.info/nitro/appmanual_v2/index.html?functions_1.html)

**Parselookupvalue**

This function is used in column mappings (create/update item) to set the value for a lookup column.

This function takes one of the following as input parameter and returns the first item matching that value.

1.Literal/custom value

a.Syntax: $parselookupvalue(LookupColumnValue)

b.Example: $parselookupvalue(Calendar)

2.Another column value (placeholder)

a.Syntax: $parselookupvalue([FieldDisplayName|FieldInternalName])

b.Example: $parselookupvalue([Title|Title])

3.Value from a Query List Action

a.Syntax: $parselookupvalue(QuerylistActionName##[FieldDisplayName|FieldInternalName])

b.Example: $parselookupvalue(ConfigurationItem##[Title|Title])

4.Value from a variable

a.Syntax: $parselookupvalue(VariableName##Value)

b.Example: $parselookupvalue(UsersVariable##Value)

**Note:** This function is only used in field mappings and the target column should be a lookup column.

**Product**

Performs product (multiplication) of two values.

**Syntax:**

Column Value:  $product(ColumnValue,ColumnValue)

Column name:  $product([FieldDisplayName|FieldInternalName], [FieldDisplayName|FieldInternalName])

Query List:  $product(QuerylistActionName##[FieldDisplayName|FieldInternalName],ColumnValue)

Variable:  $product(VariableName##Value,ColumnValue)

**Example**

Column Value - $product (25, 5)

Column name - $product([Price|Price],[Quantity|Quantity])

Query List - $product(GetPurchaseItem##[Price|Price],2)

Variable - $product(VariableName##Value,5)

**Note**

The values used in the function should be number values or should be resolved to numbers and

target column should be number or currency field

**calcdate**

Adds specific values to date time object

**Syntax**

$calcdate(FieldDisplayName|FieldInternalName),number, Days/Weeks/Months/Years)

**Example**

$calcdate([Due Date|DueDate],5,Months)

$calcdate([Due Date|DueDate],5,Days)

**Xpath**

This function is used to navigate through elements and attributes in an XML text. This function takes two arguments as input. First argument is one of the following possible parameters and returns the value as per the “xpath expression” given in second argument. Second argument is always expected to be a valid xpath expression.

First Argument possible parameters:

1.Literal/Custom Value

**Syntax:** $xpath(any valid xml text,@xpath expression)

**Example:** $xpath(<birds><parrot><color>green</color></parrot><pigeon><color>white</color></pigeon></birds>, @/birds/parrot)

2.Other column value (Place holder of current item)

**Syntax:**$xpath([placeholder],@xpath expression)

**Example:** $xpath([Cities|Cities], @/Cities)

3.Other column value (Place holder of query list action executed prior to this)

**Syntax:** $xpath(QuerylistActionName##Field placeholder,@xpath expression)

**Example:** $xpath(TicketItem##Cities, @/Cities)

4.Variable value

**Syntax:** $xpath(VariableName##Value,@xpath)

**Example:**$xpath(VarName##Value,@/country/city)

**Detailed Examples with output**

Example 1:

My xml text input for this example:

<birds>

<parrot><color>green</color></parrot>

<pigeon><color>white</color></pigeon>

</birds>

Expression: $xpath([xmlvalue|xmlvalue], @/birds/parrot)

Output: <parrot><color>green</color></parrot>

Example 2:

My input is coming from a column of Query List Action (action name: TicketItem, Cities is a column name in the query list action list) result list item. And sample input is,

<Cities>

 <City id="1">Hyd</City>

 <City id="2">Bng</City>

 <City id="3">Delhi</City>

</Cities>

Expression 1: $xpath(TicketItem##Cities, @/Cities/City)

Output 1: [{id: 1, City: Hyd},{id: 2, City: Bng},{id: 3, City: Delhi}]

Output 2: Hyd

Expression 2: $xpath(TicketItem##Cities, @/Cities)

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

Output 2: <City id="1">Hyd</City>

 <City id="2">Bng</City>

 <City id="3">Delhi</City>

Expression 3: $xpath(TicketItem##Cities, @/Cities/City/@id)

Output 1: [{id: 1},{id: 2},{id: 3}]

Output 2: 1

Note:

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.

**regextract**

This function is used to extract a pattern value from a string using regex expression

**Syntax**

$regextract([FieldDisplayName|FieldInternalName],@.expression)

**Example**

$regextract([Subject|Subject],@.\*\[CaseId: (.+)\].\*)

**split**

splits the string to a collection of strings using the separator. This can be only used in variable action.

Sample Use case: Get the approvers from a user/text field separated with an identifier like # or @ and create child items with each of these as field mappings. We can use “@{CollectionValue}” in the field mappings.

Steps:

- Define the Split function in custom action.

- Create a custom action for add item and use variable in setting and set field mapping as @{CollectionValue}

**Syntax**

$split(separator,[FieldDisplayName|FieldInternalName])

**Example**

$split(:,[Title|Title])

**Sum**

This function is used to calculate sum of the values of a column of the query list item collection result. This works only if Query List action type is “Item Collection”

**Syntax**

QuerylistActionName##$sum([FieldDisplayName|FieldInternalName])

**Example**

GetPurchaseItems##$sum([Cost|Cost])

Note: It supports only single column place holder.

**variablecollection**

This function is used to get the collection of values from a column of the query list item collection result and combines in to a single value by joining the collection of values with the given separator in second argument.

**Syntax**

QueryListActionName##$variablecollection(separator,[FieldDisplayName|FieldInternalName])

**Example**

GetDepartments##$variablecollection(;#,[Owner|Owner])

**Lookupid**

Returns given lookup/user column’s ID.

**Syntax**

$lookupid([FieldDisplayName|FieldInternalName])

**Example**

$lookupid([Requester|Requester])

**Parseuservalue**

Used in column mappings (create/update item) to set the value for a user column.  
This function will try to get user/group id from web site users/groups.

**Syntax**

$parseuservalue(User Display Name/SharePoint Group Name)

**Example**

$parseuservalue(Pavan Kumar)

$parseuservalue(Owners)

**Note:** This function can be used only in field mappings and the target field should be User field.

**Parsefieldvalue**

This function is used to return required output from field value object

Default value type: **String**

Default Separator: **; (semi colon)**

$parsefieldvalue(field placeholder)

$parsefieldvalue(field placeholder, value type)

$parsefieldvalue(field placeholder, value type, separator)

Available value types:

-**Id**

-**Value**

-**Email**

-**Dateonly**

-**url**

-**string**

-**object**

Examples:

$parsefieldvalue(field placeholder)

**Jsonpath**

This function is used to navigate through objects in an JSON object/array. This function takes two arguments as input. First argument is one of the following possible parameters and returns the value as per the “json path” given in second argument. Second argument is always expected to be a valid json path.

First Argument possible parameters:

5.Literal/Custom Value:

Syntax: $jsonpath (any valid json object,@jsonpath)

2.Other column value (Place holder of current item)

Syntax: $jsonpath ([placeholder],@jsonpath)

3.Other column value (Place holder of query list action executed prior to this)

Syntax: $jsonpath (QuerylistActionName##Field placeholder,@jsonpath)

4.Variable value

Syntax: $jsonpath (VariableName##Value,@jsonpath)

**Detailed Examples with output**

Example 1:

My JSON object input for this example:

{"MovieDatabase": {

         "movie": [{

                 "name": "The Change-Up",

                 "genre": "comedy",

                 "director": "David Dobkin",

                 "Facebook\_like": 252

         },

         {

                 "name": "Rise of the Planet of the Apes",

                 "genre": "SciFi",

                 "director": "Rupert Wyatt",

                 "Facebook\_like": 472

         }]

 }

}

Expression: $jsonpath(JSON object, @$.MovieDatabase.movie[\*].director)

Output: ["David Dobkin","Rupert Wyatt"]

Example 2:

My input is coming from a column of Query List Action (action name: TicketItem, Cities is a column name in the query list action list) result list item. And sample input is same as above,

Expression 1: $jsonpath(TicketItem##Movies, @$.MovieDatabase.\*)

Output: [[{"name":"The Change-Up","genre":"comedy","director":"David Dobkin","Facebook\_like":252},{"name":"Rise of the Planet of the Apes","genre":"SciFi","director":"Rupert Wyatt","Facebook\_like":472}]]

Expression 2: $jsonpath(TicketItem##Movies, @$.MovieDatabase..Facebook\_like)

Output: [252,472]

Expression 3: $jsonpath(TicketItem##Movies, @$..movie[(@.length-1)])

Output: [{"name": "Rise of the Planet of the Apes","genre": "SciFi","director": "Rupert Wyatt","Facebook\_like": 472}]