UiPath Activities Manual

MongoDB Custom Activities Package

-custom activities-

Table of Contents

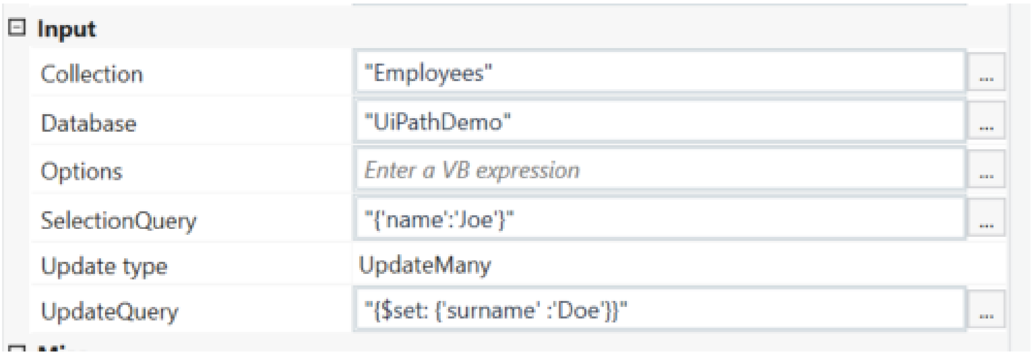
Prerequisites

The current activities package is built for the .NET 6.0 platform. It requires at least Studio version 22.10.

Query Format

The activities in the package receives JSON Object as query parameters. These parameters can be created in two ways:

Directly in string format. Example from the Update activity:

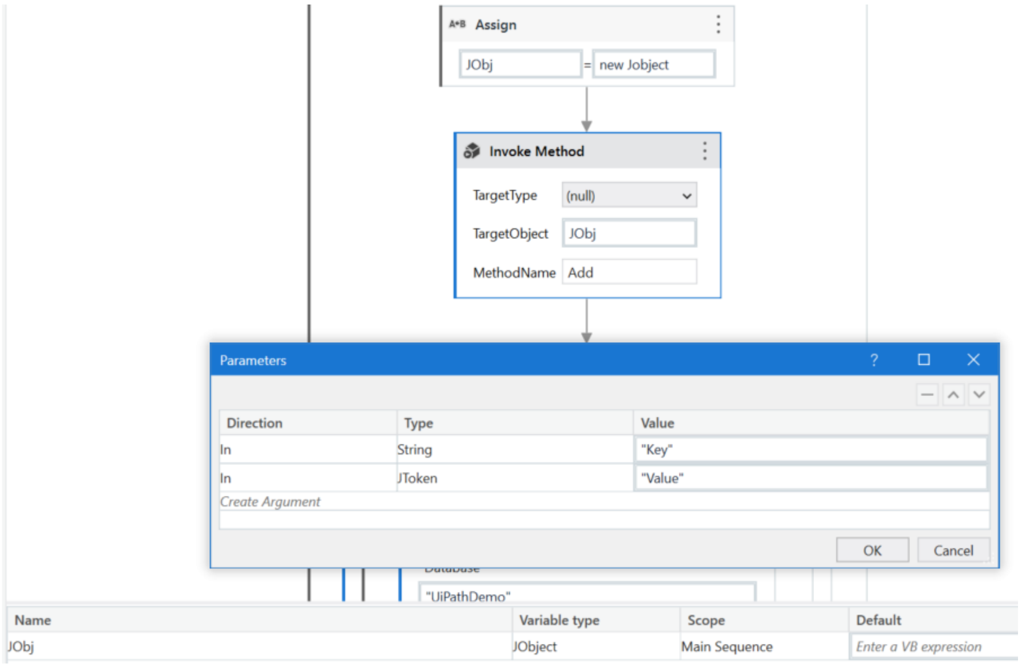


Input

Collection	"Employees"
Database	"UiPathDemo"
Options	*Enter a VB expression*
SelectionQuery	"{'name':'Joe'}"
Update type	UpdateMany
UpdateQuery	"{$set: {'surname' :'Doe'}}"

Double quotes are used for the string, and single quotes for properties names. Operators, like in the above example $set are not in between single quotes.

The second option is using JObjects stringified:



Assign: JObj = new Jobject

Invoke Method

TargetType	(null)
TargetObject	JObj
MethodName	Add

Parameters

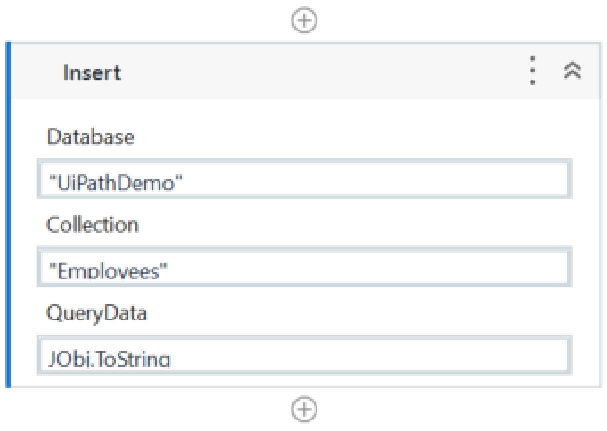
Direction	Type	Value
In	String	"Key"
In	JToken	"Value"

Create Argument

OK Cancel

Name	Variable type	Scope	Default
JObj	JObject	Main Sequence	*Enter a VB expression*

An empty JObject is created and then using the invoke method properties are added to the object. Then inside the activity, you can pass the variable transformed in a String as a parameter:


Insert

Database

"UiPathDemo"

Collection

"Employees"

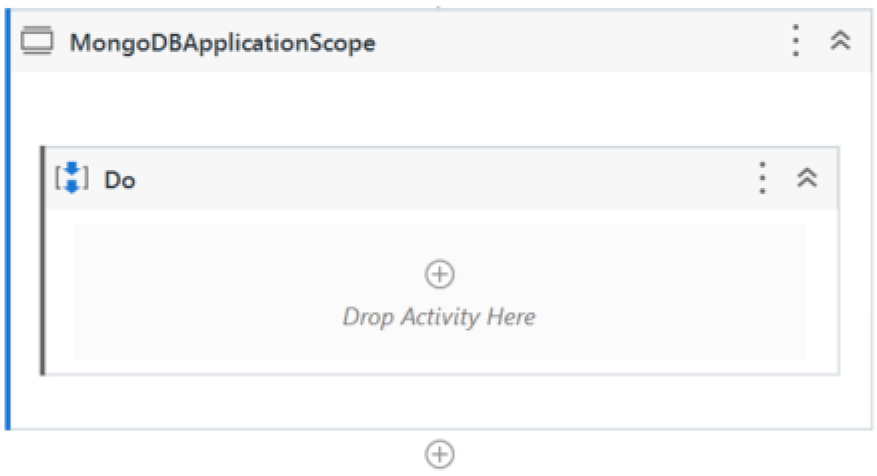
QueryData

JObj.ToString

For clarity, in the rest of the documentation the first method is used.

Activities

MongoDB Application Scope



This activity will be used as a container for all the other activities in the package. Its main purpose is to perform the connection and authentication with the Mongo Database.

Parameters

ConnectionString: The string used to connect to the database.

Example

The standard URI connection scheme has the form:
mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]

The following string is used to connect to a local instance of the database that runs on the port 27017:
"mongodb://localhost:27017"

More examples can be found on the following page:
https://www.mongodb.com/docs/manual/reference/connection-string/

Find

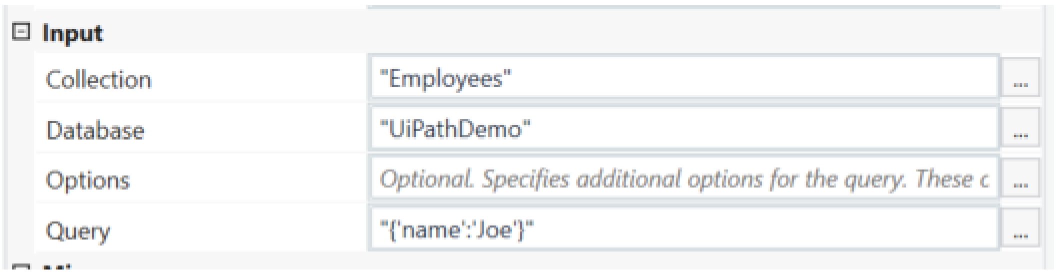
Selects documents in a collection or view and returns a cursor to the selected documents. Is the equivalent of the SQL clause Select.

Parameters

- Collection - The name of the collection from where the data is extracted.
- Database - The database from where the data is extracted.
- Options - Specifies additional options for the query. These options modify query behaviour and how results are displayed. The available options can be found here.
- Query - Specifies selection filter using query operators. The query is formatted as a JSON string.

Examples

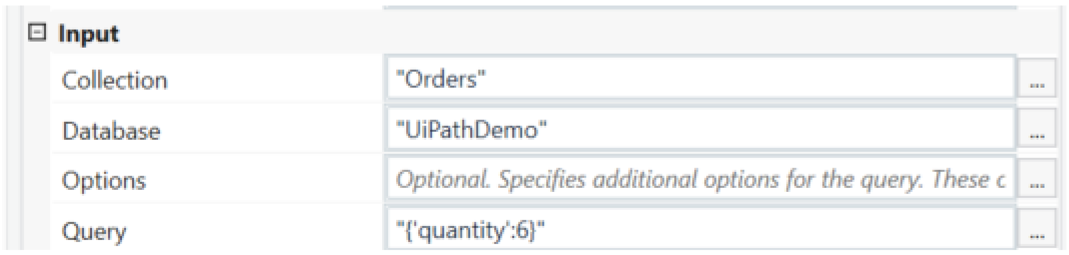
The example from below returns all the documents that have the name equal to "Joe", from the database UiPathDemo, from the collection Employees.



Input

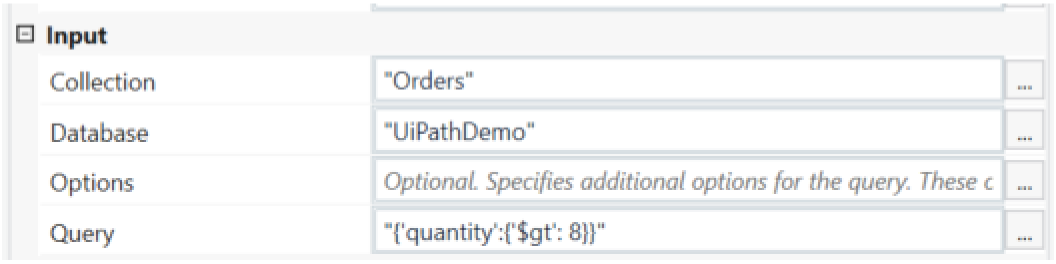
Collection	"Employees"
Database	"UiPathDemo"
Options	Optional. Specifies additional options for the query. These c
Query	"{'name':'Joe'}"

In the next two examples we can see how we can work with numerical data. In the first example we can see how the orders with a quantity equal to 6 are extracted.



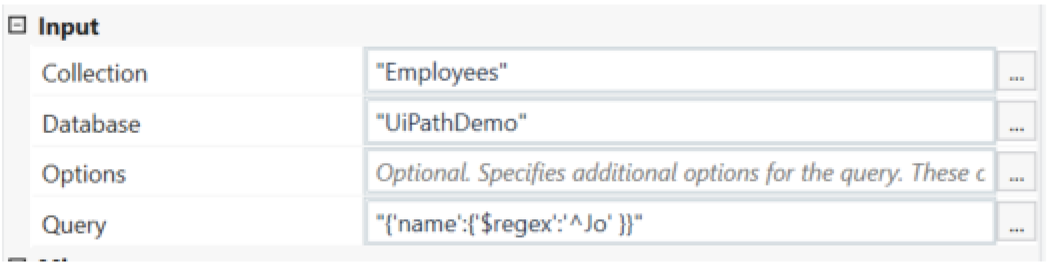
Input	
Collection	"Orders"
Database	"UiPathDemo"
Options	*Optional. Specifies additional options for the query. These c*
Query	"{'quantity':6}"

In the next example all the orders with a quantity greater than 8 are extracted. The comparison operators used are the standard MongoDB ones.



Input	
Collection	"Orders"
Database	"UiPathDemo"
Options	*Optional. Specifies additional options for the query. These c*
Query	"{'quantity':{'$gt': 8}}"

In the last example find activity is used with a regex, to extract all the employees that have the name starting with Jo. More information about regex for MongoDB can be found in the official documentation.



Input	
Collection	"Employees"
Database	"UiPathDemo"
Options	*Optional. Specifies additional options for the query. These c*
Query	"{'name':{'$regex':'^Jo' }}"

The activity implements the official mongo find activity. More advanced ways of using the find activity and more advanced queries examples can be found in the official MongoDB documentation: https://www.mongodb.com/docs/manual/reference/method/db.collection.find/

Insert

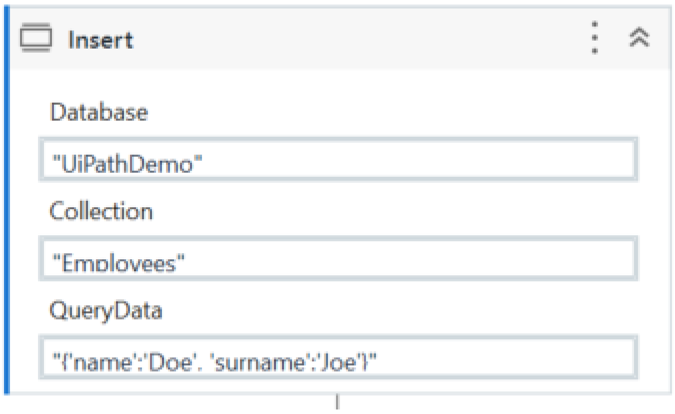
Inserts a document or documents into a collection.

Parameters

- Collection - The name of the collection where the data is added.
- Database - The database where the data is added.
- QueryData – The document or documents that will be inserted, in a string format.

Examples

The example below adds an employee with the surname John and the name Doe in the collection Employees of the UipathDemo Database:


Insert

Database

"UiPathDemo"

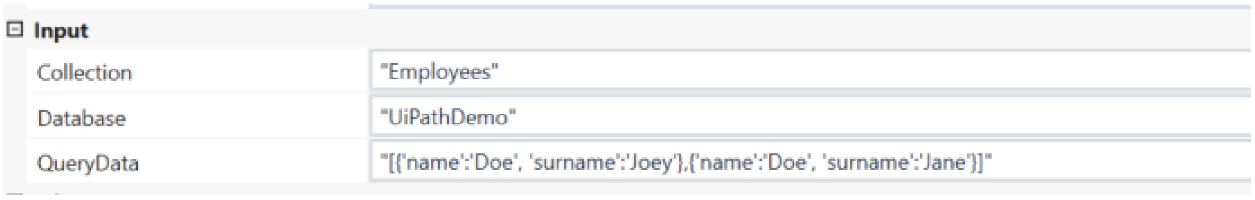
Collection

"Employees"

QueryData

"{'name':'Doe', 'surname':'Joe'}"

The QueryData parameter can also be a JSON array like in the following example, for adding multiple employees:


Input

Collection	"Employees"
Database	"UiPathDemo"
QueryData	"[{'name':'Doe', 'surname':'Joey'},{'name':'Doe', 'surname':'Jane'}]"

Update

Modifies an existing document or documents in a collection. The method can modify specific fields of an existing document or documents or replace an existing document entirely, depending on the update parameter.

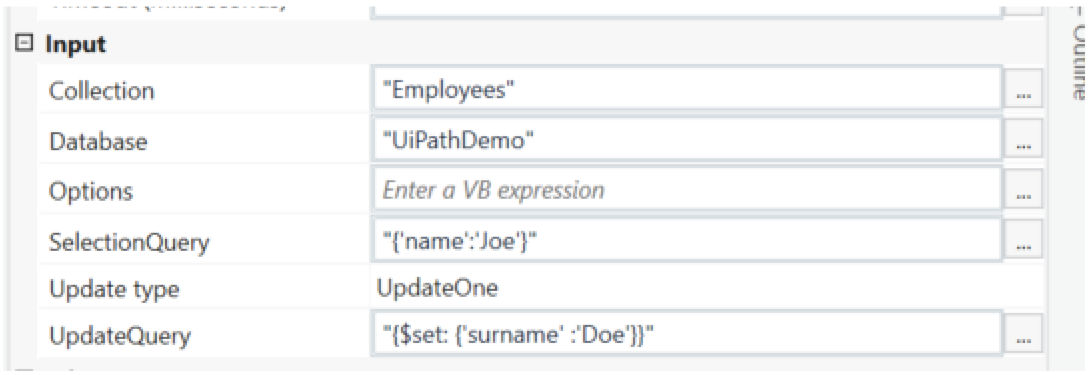
Parameters

- Collection: The name of the collection where the data is modified.
- Database: The database where the data is modified.
- Options: Optional. Specifies additional options for the update. The options are:
  - BypassDocumentValidation: Gets or sets a value indicating whether to bypass document validation.
  - IsUpsert: Gets or sets a value indicating whether to insert the document if it doesn't already exist.
  - Sort: Gets or sets the sort.
- Selection Query: The selection criteria for the update. The same query selectors as in the find activity are available.
- Update Query: The modifications to apply.
- Update Type: The type of update to be executed. The options are:

- Update One: Update the first document that matches the selection criteria.
- Update Many: Update all the documents that match the selection criteria.
- Replace One: Replaces the first document that matches the selection criteria.

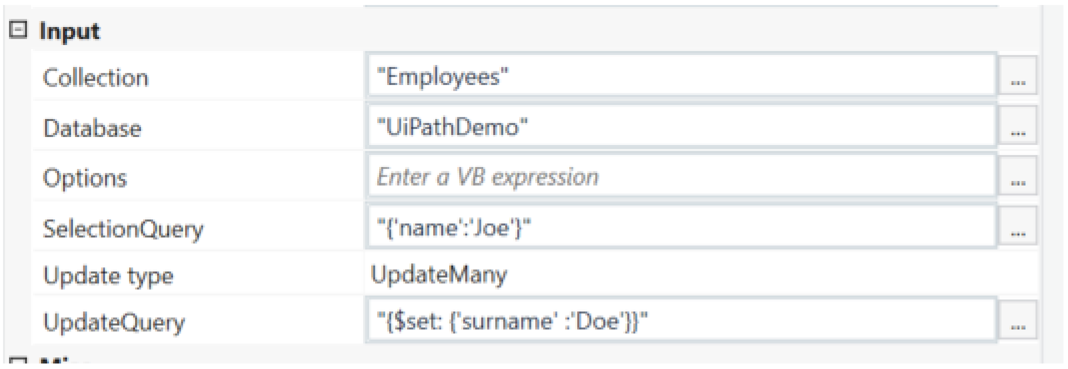
Examples

The below example updates the surname of the first employee found with the name Joe.



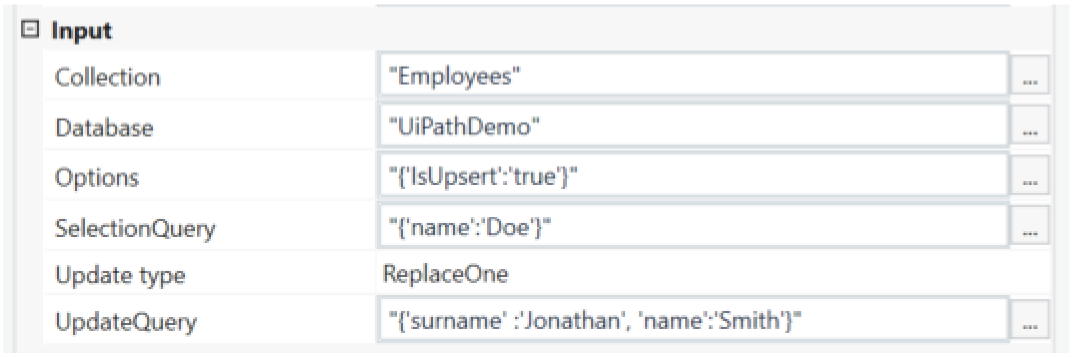
Input	
Collection	"Employees"
Database	"UiPathDemo"
Options	Enter a VB expression
SelectionQuery	"{'name':'Joe'}"
Update type	UpdateOne
UpdateQuery	"{$set: {'surname' :'Doe'}}"

Changing the Update Type makes the activity to update all the employees found with the name Joe



Input	
Collection	"Employees"
Database	"UiPathDemo"
Options	Enter a VB expression
SelectionQuery	"{'name':'Joe'}"
Update type	UpdateMany
UpdateQuery	"{$set: {'surname' :'Doe'}}"

In the next example the activity will try to replace an employee having the name Doe with Jonathan Smith. Having the option IsUpsert set to true, if the selection criteria is not found, if no one with the name Doe exists in the database, a new document will be inserted with the parameters from Update Query. If IsUpsert was set to false, or if no options would be sent as parameters, then if the selection criteria is not matched with anything then no document will be modified.



Input	
Collection	"Employees"
Database	"UiPathDemo"
Options	"{'IsUpsert':'true'}"
SelectionQuery	"{'name':'Doe'}"
Update type	ReplaceOne
UpdateQuery	"{'surname' :'Jonathan', 'name':'Smith'}"

Delete

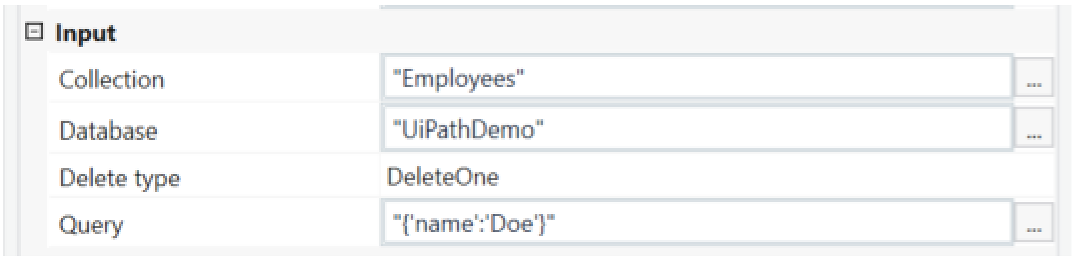
Removes one or all the documents matching the filter criteria from a collection.

Parameters

- Collection – The name of the collection from where the documents will be removed.
- Database – The name of the database where the collection is.
- Delete Type: the type of delete to be executed. The options are:
  - Delete One: Remove the first document that matches the filter criteria.
  - Delete Many: Remove all documents that match the filter criteria.
- Query: Specifies deletion criteria using query operators.

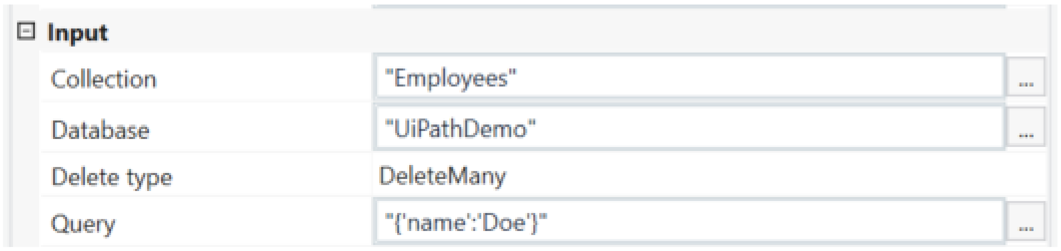
Examples

In the below example the first matching employee named Doe is removed from the database.



Input	
Collection	"Employees"
Database	"UiPathDemo"
Delete type	DeleteOne
Query	"{'name':'Doe'}"

Changing the delete type removes all the employees named Doe from the database.



Input	
Collection	"Employees"
Database	"UiPathDemo"
Delete type	DeleteMany
Query	"{'name':'Doe'}"

CreateCollection

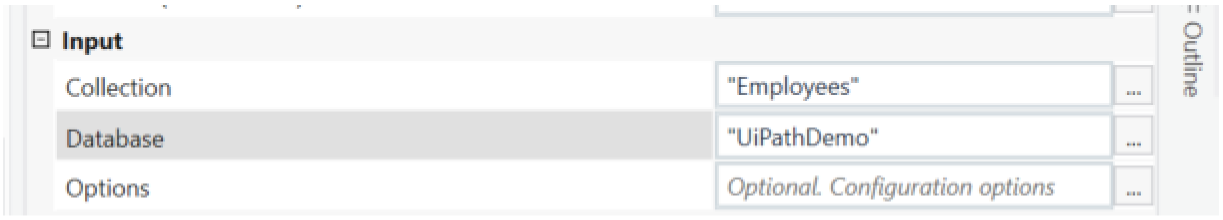
Creates a new collection or view.

Parameters

- Collection – The name of the newly created collection.
- Database – The name of the database where the new collection will be added.
- Options - Configuration options for creating a Capped collection, a Clustered collection or a View.

Example

The below example creates the collection Employees in the database UiPathDemo



Input	
Collection	"Employees"
Database	"UiPathDemo"
Options	Optional. Configuration options

All the options that can be used when creating a collection can be found in the official documentation.

DropCollection

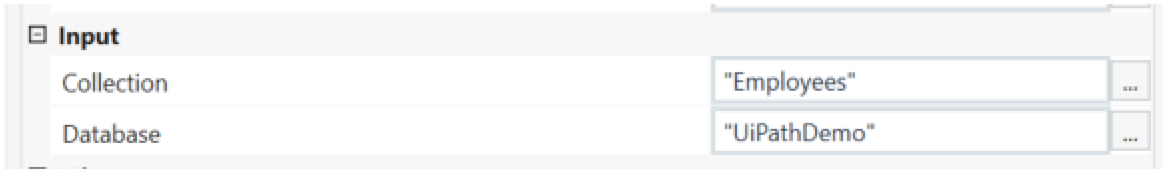
Removes a collection or view from the database. The method also removes any indexes associated with the dropped collection.

Parameters

- Collection – The name of the collection that will be removed.
- Database – The name of the database where the collection that will be removed is located.

Example

The below example drops the collection Employees from the database UiPathDemo



Input	
Collection	"Employees"
Database	"UiPathDemo"

DropDatabase

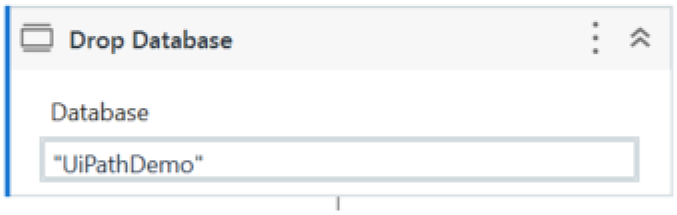
Removes a database, deleting the associated data files.

Parameters

- Database: The name of the database that needs to be dropped

Example

The below example drops the database UiPathDemo


Drop Database

Database

"UiPathDemo"

ListDatabases

The ListDatabases activity provides a list of the names of all the existing databases.

Parameters

- Databases (output) – An array containing the names of the existing databases.

ListCollections

The ListCollections activity provides a list of the names of all the names of the collections from a database.

Parameters

- Database: The database where the collections are.
- Collection Names (output): An array containing the names of the existing collections from the database.

RenameCollection

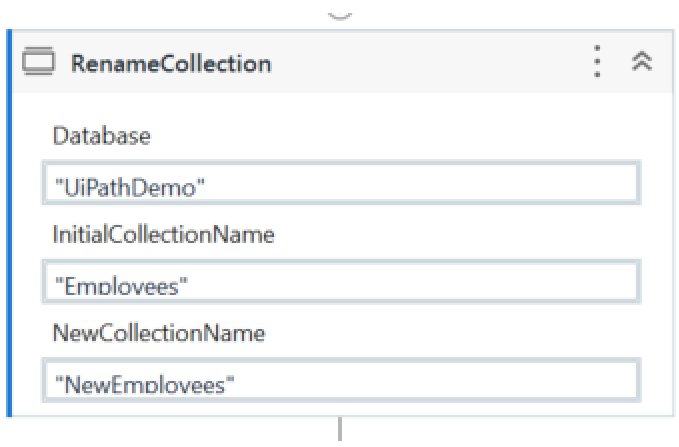
Renames a collection.

Parameters

- Database: The database where the collection is.
- InitialCollectionName: The old name of the collection.
- NewCollectionName: The new name of the collection.

Example

The example from below changes the name of the collection Employees from the database UiPathDemo to NewEmplooyees.



RenameCollection

Database
"UiPathDemo"
InitialCollectionName
"Employees"
NewCollectionName
"NewEmployees"

Estimated Count

Returns the count of all documents in a collection or view.

Parameters

- Database: The database where the collection is located.
- Collection: The collection that will be queried.
- Number Of Documents (output): The number of documents from the collection.

Run Command

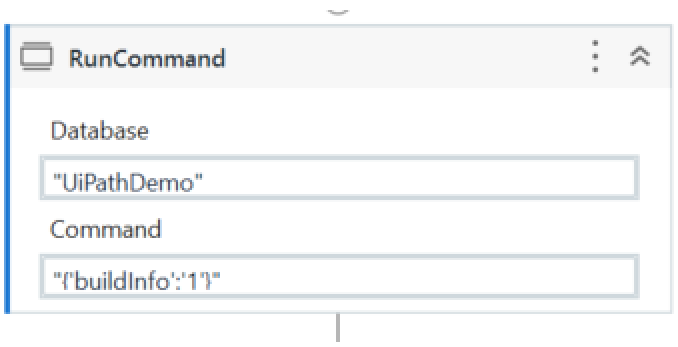
Provides a helper to run specified database commands.

Parameters

- Database: The name of the database.
- Command: The command to be executed.
- Options: Optional. Additional options for the command.
- Result (Output): The result of running the command.

Example

The following command extracts statistics about the MongoDB build and saves them in the output variable. A list with all the commands can be found in the official documentation.


RunCommand

Database

"UiPathDemo"

Command

"{'buildInfo':'1'}"

Aggregate

Calculates aggregate values for the data in a collection or a view.

Parameters

- Collection: Name of the collection where the documents are.
- Database: Name of the database where the collection is.
- Pipeline: A sequence of data aggregation operations or stages.
- Options: Optional. Additional options passed to the aggregate activity. Available only if you specify the pipeline as an array.

Example

The following example uses a collection containing the following document:

[{ "_id": 1, "cust_id": "abc1", "status": "A", "amount": 50 },

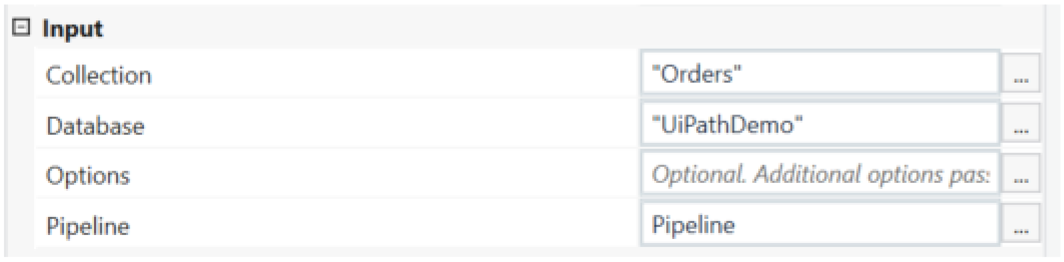
{ "_id": 1, "cust_id": "xyz1", "status": "A", "amount": 100 },

{ "_id": 1, "cust_id": "xyz1", "status": "D", "amount": 25 },

{ "_id": 1, "cust_id": "xyz1", "status": "D", "amount": 125 },

{ "_id": 1, "cust_id": "abc1", "status": "A", "amount": 25 }]

The following aggregation operation selects documents with status equal to "A", groups the matching documents by the cust_id field and calculates the total for each cust_id field from the sum of the amount field, and sorts the results by the total field in descending order:



Input

Collection	"Orders"
Database	"UiPathDemo"
Options	Optional. Additional options pas:
Pipeline	Pipeline

Where the variable Pipeline value is: "[{ '$match': { 'status': 'A' } }, { '$group': {'_id': '$cust_id', 'total': { '$sum': '$amount' } } },{'$sort': { 'total': -1 } }]"

The operation returns an array with two elements:

{ "_id" : "xyz1", "total" : 100 }

{ "_id" : "abc1", "total" : 75 }