

# Updating MongoDB Documents



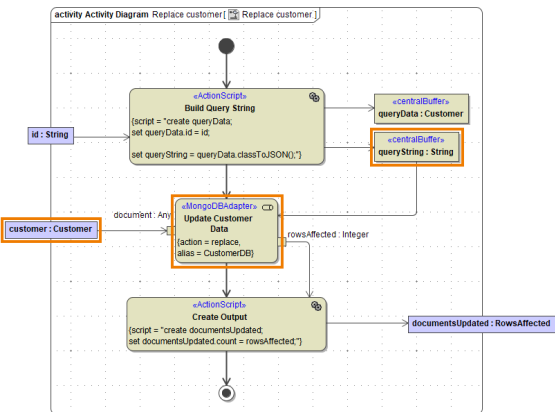
This page explains the **MongoDB Adapter** in Bridge context. If you were looking for the same information regarding the **PAS Designer**, refer to **MongoDB Adapter** in the Designer guide.

Use stereotype `<<MongoDBAdapter>>` on an action node to interact with a MongoDB and to insert, get and manipulate documents.

## Example File (Builder project Add-ons/MongoDB):



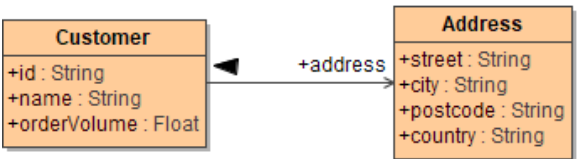
<your example path>\Add-ons\MongoDB\uml\simpleMongoDbAccess.xml



To update documents in a MongoDB database, you can use the **update** and **replace** actions. The example above shows a **replace** action. Parameter **queryString** identifies the document to be replaced, **document** supplies the new document.

## Finding the Documents to Update

For all actions that refer to existing documents, you need to provide a query string (**queryString**) to identify them. A query string contains all properties of the document you want to use for selection. The simplest way to create a query string is to create an object having the structure of the document (**queryData** in the example above), and set all query values to this object.



Then, provide this object as **queryString** by converting it to JSON using `classToExtendedJSON()`.

## Updating Data

You can update complete documents or only parts of them, depending on the input you give to the adapter.


Action	Name	Type	Description
--------	------	------	-------------

### On this Page:

- Finding the Documents to Update
- Updating Data
  - Building an Update String
  - Updating a Document Via an Update String
  - Updating a Document Using Parameter "Document"
  - Replacing a Document

### Related Pages:

- MongoDB Components
- Querying MongoDB
- Updating MongoDB Documents
- Aggregating Data
- Inserting and Deleting Documents
- MongoDB Adapter Reference

update	updateString	String	<p>MongoDB update string as valid JSON.</p> <div>  In contrast to the MongoDB shell, the JSON keys must be quoted properly. </div>
	document	Any <document class>	<p>A data structure that contains the new document data. For this, the following rules apply:</p> <ul style="list-style-type: none"> <li>unset top-level fields will be skipped</li> <li>complex fields will be replaced entirely</li> </ul>
replace	document	Any <document class>	<p>A data structure representing the document you want to replace the selected document with. The old stored document will be replaced by the new one, the MongoDB <code>_id</code>, however, will not change.</p>

The following examples refer to the same MongoDB document structure as [shown above](#).

## Building an Update String


Assume you have a customer database containing the following customer document:

```
{
  "name": "John Snow",
  "company": "Winter & Partners",
  "address": {
    "street": "99, Malamute Street",
    "city": "Anchorage, AK 99506",
    "country": "USA"
  }
}
```

The changes that should be applied are propagated to MongoDB via an update string. Either you yourself build an update string to be used with an **update** adapter action, or the xUML Runtime translates a document object you have provided into an update string.

<pre>{   "\$set": {     "name": "John Snowflake"   } }</pre>	<p>A valid update string contains a <b>\$set</b> key providing as value the changes to be applied.</p>
--	--

To build an update string, we recommend to not use `concat()` operations but to create a data structure that represents the update string and can be converted to JSON with `classToExtendedJSON()`.

 Building an update string manually (e.g. using `concat()`) is susceptible to code injection.

Find below the class structure that represents an update string for the customers example.

<div> <div>UpdateCustomer</div> <div>«E2Attribute»+setOperator : Customer{externalName = "\$set"}</div> </div>	<ul style="list-style-type: none"> <li>As a class property cannot have a name <code>\$set</code>, you need to apply stereotype <code>&lt;&lt;E2Attribute&gt;&gt;</code> and an external name <b>\$set</b>.</li> <li>The structure below the <b>\$set</b> key reflects the document structure (<b>Customer</b>) of a customer.</li> </ul> <p>To set values within the update string, create an object of type <b>UpdateCompleteAddress</b> and assign values to the properties you want to change.</p>
--	---

If you want to update parts of a sub-structure, e.g. the customer address, you need to be careful. Have a look at the following example:

Currently in Database	Update String	New in Database
<pre>{   "name":   "John   Snow",    "company":   "Winter &amp;   Partners",    "address":   {      "street":     "99,     Malamute     Street",      "city":     "Anchorage     , AK     99506",      "country":     : "USA"   } }</pre>	<div>1</div> <div><div>UpdateCustomer</div><div>«E2EAttribute»+setOperator : Customer{externalName = "\$set"}</div></div> <div><pre>{   "\$set": {     "address": {       "city": "Dallas, TX 75043"     }   } }</pre></div>	<pre>{   "Name":   "John   Snow"    "company":   :   "Winter   &amp;   Partners",    "address":   : {      "city":     "Dallas,     TX 75043"   } }</pre>
	<div>2</div> <div><div>UpdateAddressProperties</div><div>▲</div><div>«E2EAttribute» +setOperator {externalName = "\$set"}</div><div>AddressProperties</div><div>«E2EAttribute»+street : String(externalName = "address.street") «E2EAttribute»+city : String(externalName = "address.city") «E2EAttribute»+postcode : String(externalName = "address.postcode") «E2EAttribute»+country : String(externalName = "address.country")</div></div> <div><pre>{   "\$set": {     "address.city": "Dallas, TX 75043"   } }</pre></div>	<pre>{   "Name":   "John   Snow",    "company":   :   "Winter   &amp;   Partners",    "address":   : {      "street":     "99,     Malamute     Street",      "city":     "Dallas,     TX 75043"      "country":     : "USA"   } }</pre>

2

UpdateAddressProperties

▲

«E2EAttribute»  
+setOperator {externalName = "\$set"}

▼

AddressProperties

«E2EAttribute»+street : String{externalName = "address.street"}  
«E2EAttribute»+city : String{externalName = "address.city"}  
«E2EAttribute»+postcode : String{externalName = "address.postcode"}  
«E2EAttribute»+country : String{externalName = "address.country"}

```
{
  "$set": {
    "address.city": "Dallas, TX 75043"
  }
}
```

Using update string 1 you will update the complete address. All properties of address that are not listed in the statement will be removed from the target document.

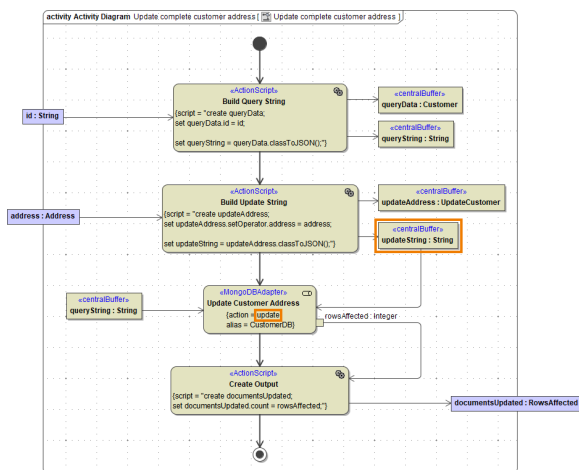


This is also the way the Runtime will translate any given update document.

Using update string 2 only the **city** property of the target document will be updated. All other properties will be left intact.

## Updating a Document Via an Update String

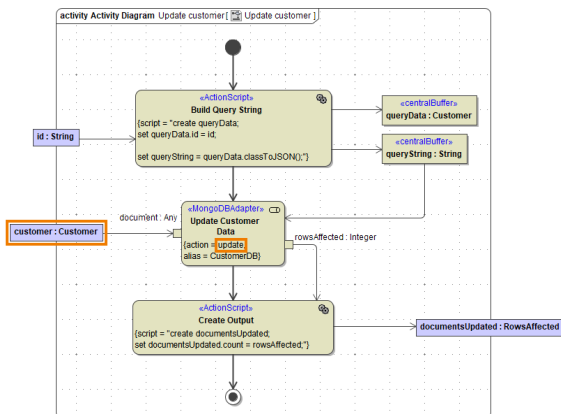
Using action **update** with the MongoDB adapter with parameter **updateString**, you can update all or dedicated properties in the selected document.



In this example, the address part of a customer document is updated. Depending on the structure of the update string, the complete address is replaced, or single address properties are updated. Refer to [Building an Update String](#) for more information on how to build an update string, and an example of the implications.

## Updating a Document Data Using Parameter "Document"

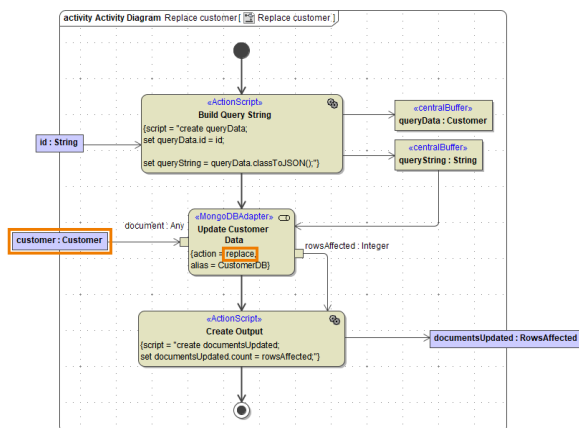
Using action **update** with the MongoDB adapter with parameter **document**, you can update all or dedicated properties in the selected document.



The provided document will be translated to an update string by the xUML Runtime. In this case, the same rules apply as for updates via update string, [especially the rule concerning properties of complex structures within a document](#).

## Replacing a Document

Using action **replace** with the MongoDB adapter you can replace a complete document in the database.



The document provided with the parameter replaces the document indicated by the query string as is. The MongoDB `_id`, however, stays intact.



The structure of the old document does not necessarily need to match the structure of the new document.