ODBapi

`ODBapi[com → "add<Command>", options]`
Uses the POST method of the HTTP protocol for both reading and writing the database

`ODBapi[com → "del<Command>", options ]`
Uses either the POST or the DELETE method of the HTTP protocol to destructively alter the database

`ODBapi[com → "upd<Command>", options]`
Uses either the POST HTTP method to update record values with SQL UPDATE command or the PUT and PATCH methods of the HTTP protocol to update OrientDB structured Document records

`ODBapi[com → "get<Command>", options ]`
Uses the GET method of the HTTP protocol to retrieve values from the database. Operations are idempotent, i.e. they do not alter the database

`ODBapi[com → "import/export", options ]`
Export a gzip file that contains the database JSON export using the GET method. Import a database from an uploaded JSON text file.

`ODBapi[com → "login/logout", options ]`
Uses the GET method of the HTTP protocol to connect to a remote server using basic authentication and the same method for disconnecting

MORE INFORMATION

- **Details and Options**

URL Construction

- The following options are used in the construction of the HTTP request.
  The OrientDB RESTful API uses the same syntax for all HTTP methods.
- **Syntax: http://<server>:<port>/<command>/[<database>/<arguments>]**
- Default options for the OrientDB server → "localhost" and the port → "2480"
- Default option for the **<database>**, db → ""
- Default option for the **<command>,** com → ""
- Default options for **<arguments>**. These are controlled with the arg → "" option.
  This option value is dependent on the following options :
- dbtype → "" option is used in addDatabase command
  and typical values are "plocal/graph", "plocal/document"
- class → "" option is used in
  addClass, addClassViaHTTP, addProperty, addPropertyViaHTTP, getClass
- propnam → "", propval → "", and proptype → "" options are used in :
  commands updPropertyValues, updPropertyValue, addPropertyViaHTTP
- id → "", with this option we pass the record id, OrientDB **@rid**. It is used in :
  updPropertyValue, updRecordPUT, updRecordPATCH, delRecords commands.

- con → "", with this option we pass the connection id used in :
  delConnection command

Control of the URLFetch command

- Each ODBapi command is executed through URLFetch and by default it returns
  from OrientDB server the contents as a JSON string and a status code.
- Default option for the HTTP method to use for request, method → "POST"
- Default option for parameters to be sent for the http request, param → {}
  URL parameters are not used in OrientDB API.
- Default option for the contents of message body to be sent, body → ""
  The body option is used indirectly to pass the sql script and sql commands.
  The following options are used to constuct the message body:
- sql → "" option is used in addOSQLScript, addOSQLCommand, getOSQLCommand
- class → "" option is used in :
  addDOCUMENT, addCONTENT, addVALUES, addClass, addPropertyValues, updPropertyValues,
  delClass, delAllRecords, delProperty, delPropertyValues, getClass
- class → "" and superclass → "" options are used in addClass command
- keys → "" and values → "" options are used in addVALUES command
  These options are formatted according to SQL-92 syntax (see an example)
- from → "" and to → "" options are used in creating edges with content, addCONTENT command.
- construct → "" is used in addCONTENT and addVALUES commands
- record → "" option takes a JSON string and is used in
  addDOCUMENT, addCONTENT, updRecordPUT, updRecordPATCH,
- ver → -1 option controls the version of the record to update, it is used in
  updRecordPUT, updRecordPATCH,
- attribnam → "", attribval → "" options are used in updDatabase, updClass, updProperty
- Default options for connecting to the server with usr → "admin" and pwd → "admin"

Enumerated Options

□ **debug**	True/False If True, prints message body of the http request
□ **com**	addOSQLScript, addOSQLCommand, addDatabase, addClass, addClassViaHTTP, addProperty, addPropertyViaHTTP, addDOCUMENT, addCONTENT, addVALUES delDatabase, delClass, delProperty, delPropertyValues, delAllRecords, delRecords, delConnection, updDatabase, updClass, updProperty, updPropertyValues, updPropertyValue, updRecordPUT, updRecordPATCH, getServer, getOSQLCommand, getDatabases, getDatabase, getClass, getRecord, importDatabase, exportDatabase, logout, login
□ **construct**	RECORD, VERTEX, EDGE
□ **method**	PUT, GET, DELETE, PUT, PATCH

- The Input Assistant of the Wolfram Predictive Interface offers context-sensitive autocompletion for the enumerated option arguments of `ODBapi`. A drop-down list with option values is filtered automatically as you start typing the name of the option value e.g. com → log... and only two option values are displayed logout and login. If you want the drop-down list filtering to appear do not start quoting the option value. You can also select any of the ODBapi function templates to assist you in the completion of the command.

- The auto-completion rules are automatically loaded on Front End from **OptionValues folder** at:

```
In[1]:= FileNameJoin[{$InstallationDirectory, "SystemFiles", "FrontEnd",
         "SystemResources", "FunctionalFrequency", "OptionValues"}]
Out[1]= C:\Mathematica\SystemFiles\FrontEnd\SystemResources\FunctionalFrequency\OptionValues
```

- Therefore to enable auto-completion for the ODBapi function search for the file "ODBapi.m". If you installed the package under the $UserBaseDirectory then it is located at:

```
In[2]:= $UserBaseDirectory <> "\\Applications\\DBAPI\\Options\\ODBapi.m"
Out[2]= C:\Users\athanassios\AppData\Roaming\Mathematica\Applications\DBAPI\Options\ODBapi.m
```

- Copy that file inside the **OptionValues folder** of your *Mathematica* installation.

Change Default Options

- Default options can be changed with the SetOptions command e.g.

```
In[1]:= SetOptions[ODBapi, db → "DemoDB"];
```

EXAMPLES

Basic Examples (1)

Load the two packages that are included in the **DBAPI`** context, the **DBAPI`Utils`** and the **DBAPI`OrientDB`**:

```
In[1]:= << DBAPI`
```

```
Version → 10.2.0 for Microsoft Windows (64-bit) (July 7, 2015)

Data Utilities Package v0.9
(c) December 2015, By Athanassios I. Hatzis

OrientDB HTTP API Package v0.9
(c) December 2015, By Athanassios I. Hatzis
```

Add Commands (5)

Add Database (1)

Add Database command creates a, disk-based or memory, document or graph, database with a username and password on remote database Server

```
In[1]:= ODBapi[com → "addDatabase", db → "DemoDB",
          dbtype → "plocal/graph", usr → "root", pwd → "123", debug → True] // Short
```

```
http://localhost:2480/database/DemoDB/plocal/graph
=== Body ===
```

```
Out[1]//Short= {{"classes":[{"name":"OUser","superClass":"OIdentity","alias":nu
                 …alue":-1}],"properties":[{"name":"strictSql","value":"true"}]}}, 200}
```

Add Class (1)

Add a class by executing SQL CREATE CLASS to create a new class in the schema and optionally extend a superclass.

```
In[1]:= ODBapi[com → "addClass", db → "DemoDB", class → "Person", debug → True];
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
CREATE CLASS Person
```

```
In[2]:= ODBapi[com → "addClass", db → "DemoDB", class → "Employee", superclass → "Person", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
CREATE CLASS Employee extends Person
```

```
Out[2]= {{"result":[{"@type":"d","@version":0,"value":13}]}, 200}
```

Add a class via HTTP

```
In[3]:= ODBapi[com → "addClassViaHTTP", db → "DemoDB", class → "Company", debug → True]
```

```
http://localhost:2480/class/DemoDB/Company
=== Body ===
```

```
Out[3]= {14, 201}
```

Add Property (1)

Add a property via SQL CREATE PROPERTY command

```
In[1]:= ODBapi[com → "addProperty", db → "DemoDB", class → "Person",
         propnam → "personName", proptype → "STRING", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
CREATE PROPERTY Person.personName STRING
```

```
Out[1]= {{"result":[{"@type":"d","@version":0,"value":1}]}, 200}
```

Add a property via HTTP

```
In[2]:= ODBapi[com → "addPropertyViaHTTP", db → "DemoDB",
         class → "Person", propnam → "personGender", proptype → "STRING", debug → True]
```

```
http://localhost:2480/property/DemoDB/Person/personGender/STRING
=== Body ===
```

```
Out[2]= {2, 201}
```

Add Records (1)

Add a JSON stuctured Document by inserting a schemaless record into Person Class via the message body

In Wolfram language we can represent the record with a hierarchical List of Rule

```
In[1]:= john = {
           "telephones" → Thread[{"home", "business", "mobile"} → {"2104566345", "2108856844", "6974059256"}],
           "firstName" → "John", "lastName" → "Brown", "DOB" → "1971-10-01", "age" → 44};
```

Then we transform the list of rules above to a JSON string using the DBexpressionToJSON function of the **Utilities Package.**

```
In[2]:= johnJSON = john // DBexpressionToJSON[#, compact → True] &
```

```
Out[2]= {"telephones":{"home":"2104566345","business":"2108856844","mobile":"6974059256"},"firstName":"John","
          lastName":"Brown","DOB":"1971-10-01","age":44}
```

```
In[3]:= ODBapi[com → "addDOCUMENT", db → "DemoDB", class → "Person", record → johnJSON, debug → True]
```

```
http://localhost:2480/document/DemoDB
=== Body ===
{"@class":"Person","telephones":{"home":"2104566345","business":"
    2108856844","mobile":"6974059256"},"firstName":"John","lastName":"
    Brown","DOB":"1971-10-01","age":44}
```

```
Out[3]= {{"@type":"d","@rid":"#12:2","@version":1,"@class":"Person","telephones":{"home":"2104566345","
          business":"2108856844","mobile":"6974059256"},"firstName":"John","lastName":"Brown","DOB":"1971-
          10-01","age":44}, 201}
```

Add the above JSON stuctured Document into Person via SQL INSERT INTO command

```
In[4]:= ODBapi[com → "addCONTENT", db → "DemoDB",
         class → "Person", record → johnJSON, construct → "RECORD", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
INSERT INTO Person content
    {"telephones":{"home":"2104566345","business":"2108856844","mobile":"
    6974059256"},"firstName":"John","lastName":"Brown","DOB":"1971-10-01"
    ,"age":44}
```

```
Out[4]= {{"result":[{"@type":"d","@rid":"#12:3","@version":1,"@class":"Person","telephones":{"home":"
    2104566345","business":"2108856844","mobile":"6974059256"},"firstName":"John","lastName":"Brown",
    "DOB":"1971-10-01","age":44}]}, 200}
```

Add the above JSON stuctured Document into Person via SQL INSERT VALUES command

```
In[5]:= kstr = john // Association // Keys // DBListSetToSQL92[#, values → False] &
```

```
Out[5]= (telephones, firstName, lastName, DOB, age)
```

We apply a series of transformation to get the Keys and Values strings in SQL-92 format. Association and Keys are Wolfram functions, `DBListSetToSQL92` is a function of the **Utilities package**.

```
In[6]:= vstr = john // Association // Values // DBListSetToSQL92
```

```
Out[6]= ({"home":"2104566345","business":"2108856844","mobile":"6974059256"},
          "John", "Brown", "1971-10-01", 44)
```

```
In[7]:= ODBapi[com → "addVALUES", db → "DemoDB", class → "Person",
         keys → kstr, values → vstr, construct → "RECORD", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===

 INSERT INTO Person (telephones, firstName, lastName, DOB, age) VALUES
   ({"home":"2104566345","business":"2108856844","mobile":"6974059256"},
    "John", "Brown", "1971-10-01", 44)
```

```
Out[7]= {{"result":[{"@type":"d","@rid":"#12:4","@version":1,"@class":"Person","telephones":{"business":"
    2108856844","mobile":"6974059256","home":"2104566345"},"firstName":"John","lastName":"Brown","DOB
    ":"1971-10-01","age":44}]}, 200}
```

Add OSQL (1)

AddOSQLScript means that the database may change. OrientDB batch of SQL commands can be non-idempotent and are executed via POST in a single call

```
In[1]:= osqlScript = "
CREATE CLASS Car EXTENDS V;
CREATE VERTEX Car SET brand='FIAT',model='Punto',year='2000-01-01'";
ODBapi[com → "addOSQLScript", db → "DemoDB", sql → osqlScript, debug → True]
```

```
http://localhost:2480/batch/DemoDB
=== Body ===
{
    "transaction": false,
    "operations": [
        {
            "type": "script",
            "language": "sql",
            "script": "\nCREATE CLASS Car EXTENDS V;\nCREATE
   VERTEX Car SET brand='FIAT',model='Punto',year='2000-01-01'"
        }
    ]
}
```

```
Out[1]= {{"result":[{"@type":"d","@rid":"#11:0","@version":1,"@class":"Car","brand":"FIAT","model":"Punto","
   year":"2000-01-01"}]}, 200}
```

AddOSQLCommand means that the database may change.
OrientDB SQL command executed under the hood via POST can be non-idempotent

```
In[2]:= ODBapi[com → "addOSQLCommand", db → "DemoDB", sql → "SELECT FROM Car", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
SELECT FROM Car
```

```
Out[2]= {{"result":[{"@type":"d","@rid":"#11:0","@version":1,"@class":"Car","brand":"FIAT","model":"Punto","
   year":"2000-01-01"}]}, 200}
```

Del Commands (5)

Del Database (1)

Delete a Database with basic authentication to the server

```
In[1]:= ODBapi[com → "delDatabase", db → "TestDB", usr → "root", pwd → "123", debug → True]
```

```
http://localhost:2480/database/TestDB
=== Body ===
```

```
Out[1]= {, 204}
```

Del Class (1)

Delete a Class, i.e. remove completely the class from the schema. If the class extends vertex (V) then all the vertices have to be deleted first. Command delClass is based on SQL DROP CLASS operation.

```
In[1]:= ODBapi[com -> "delClass", db → "DemoDB", class → "Company", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
DROP CLASS Company
```

```
Out[1]= {{"result":[{"@type":"d","@version":0,"value":true}]}, 200}
```

Del Property (1)

Delete a property via SQL - DROP PROPERTY
Be aware that although the property is removed from the schema, it still remains as a key in records that have been created with that property.

```
In[1]:= ODBapi[com → "delProperty", db → "DemoDB", class → "Person", propnam → "personGender", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
DROP PROPERTY Person.personGender
```

```
Out[1]= {, 204}
```

Delete property values - This command removes a field from all records by executing SQL UPDATE

```
In[2]:= ODBapi[com → "delPropertyValues", db → "DemoDB", class → "Person", propnam → "telephones", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
UPDATE Person remove telephones
```

```
Out[2]= {{"result":[{"@type":"d","@version":0,"value":3}]}, 200}
```

Del Records (1)

Delete records via SQL - TRUNCATE RECORD by listing all the record IDs

```
In[1]:= ODBapi[com → "delRecords", db → "DemoDB", class → "Person", id → "[12:2, 12:3]", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
TRUNCATE RECORD [12:2, 12:3]
```

```
Out[1]= {{"result":[{"@type":"d","@version":0,"value":2}]}, 200}
```

Delete All records via SQL - TRUNCATE CLASS
This command acts at a lower level than SQL DELETE Command. It cannot delete a vertex or edge class if it contains records.

```
In[2]:= ODBapi[com → "delAllRecords", db → "DemoDB", class → "Person", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
TRUNCATE CLASS Person
```

```
Out[2]= {{"result":[{"@type":"d","@version":0,"value":3,"@fieldTypes":"value=l"}]}, 200}
```

Del Connection (1)

Delete Server Connection - It requires a connection id and root password to kill the connection

```
In[1]:= ODBapi[com → "delConnection", con → "49", usr → "root", pwd → "123", debug → True]
```

```
http://localhost:2480/connection/kill/49
=== Body ===
```

```
Out[1]= {, 204}
```

Get Commands (5)

Get Server (1)

Get Server Information

In[1]:=
```
ODBapi[com → "getServer", usr → "root", pwd → "123", debug → True] // ODBgetDataset
```

```
http://localhost:2480/server
=== Body ===
```

Out[1]=

connections	{ <| connectionId → 65, remoteAddress → /127.0.0.1:49386, ⋯14 |>, ⋯4 }
dbs	{ }
storages	{ <| name → demodb, type → OLocalPaginatedStorage, path → C:/orientdb217/databases/demodb |>, <| name → DemoD
properties	{ <| name → db.pool.min, value → 1 |>, <| name → db.pool.max, value → 50 |>, <| name → profiler.enabled, value → true
globalProperties	{ <| key → environment.dumpCfgAtStartup, description → Dumps the configuration at application startup, value → …, defa

3 levels | 5 rows

The postfix operation of **ODBgetDataset** transforms JSON Output above into a Wolfram Dataset. Further processing can be applied on the resulting Dataset.

In[2]:=
```
%["connections", All, {"connectionId", "remoteAddress", "db", "user"}]
```

Out[2]=

connectionId	remoteAddress	db	user
65	/127.0.0.1:49386	–	–
39	/127.0.0.1:49357	–	–
40	/127.0.0.1:49357	–	–
41	/127.0.0.1:49357	–	–
29	/0:0:0:0:0:0:0:1:49343	DemoDB	admin

2 levels | 5 rows

Get Database (1)

Get Database Information

In[1]:=
```
dbInfo = ODBapi[com → "getDatabase", db → "DemoDB", debug → True] // ODBgetDataset
```

```
http://localhost:2480/database/DemoDB
=== Body ===
```

Out[1]=

server	<| version → 2.1.7, build → 2.1.x@r${buildNumber}; 2015-12-08 16:49:51+0000, osName → Windows 7, osVersion → 6.1, os
classes	{ <| name → CLA, superClass → , superClasses → …, ⋯7 |>, ⋯14 }
clusters	{ <| id → 0, name → internal, records → 3, conflictStrategy → …, ⋯4 |>, ⋯15 }
currentUser	admin
indexes	{ <| name → OUser.name, configuration → … |>, ⋯2 }
config	<| values → { <| name → dateFormat, value → … |>, <| name → dateTimeFormat, value → … |>, ⋯8 }, ⋯1 |>

6 levels | 6 rows

The postfix operation of **ODBgetDataset** transforms JSON Output above into a Wolfram Dataset Further processing can be applied on the resulting Dataset.

In[2]:=
```
dbInfo["classes", All, {"name", "superClass", "records", "properties"}]
```

Out[2]=

name	superClass	records	properties
CLA		1	KeyAbsent
Car	V	1	KeyAbsent
E		0	KeyAbsent
Employee	Person	0	{<| name → personDOB, linkedClass → KeyAbsent, type → DATE, ⋯6 |>, ⋯2}
OFunction		0	{<| name → idempotent, linkedClass → KeyAbsent, type → BOOLEAN, ⋯6 |>, ⋯4}
OIdentity		6	KeyAbsent
ORIDs		0	KeyAbsent
ORestricted		0	{<| name → _allowDelete, linkedClass → OIdentity, type → LINKSET, ⋯7 |>, ⋯3}
ORole	OIdentity	3	{<| name → mode, linkedClass → KeyAbsent, type → BYTE, ⋯6 |>, ⋯3}
OSchedule		0	{<| name → function, linkedClass → OFunction, type → LINK, ⋯7 |>, ⋯6}
OTriggered		0	KeyAbsent
OUser	OIdentity	3	{<| name → name, linkedClass → KeyAbsent, type → STRING, ⋯6 |>, ⋯3}
Person		0	{<| name → personDOB, linkedClass → KeyAbsent, type → DATE, ⋯6 |>, ⋯2}
V		1	KeyAbsent
_studio		1	KeyAbsent

4 levels | 15 rows

Get Databases - A list of the databases on the server

In[3]:=
```
ODBapi[com → "getDatabases", debug → True]
```

```
http://localhost:2480/listDatabases
=== Body ===
```

Out[3]=
```
{{"@type":"d","@version":0,"databases":["GratefulDeadConcerts","demodb","DemoDB"],"@fieldTypes":"
    databases=e"}, 200}
```

Get Class (1)

Get Class Information from the server

In[1]:=
```
classInfo = ODBapi[com → "getClass", db → "DemoDB", class → "Person", debug → True]
```

```
http://localhost:2480/class/DemoDB/Person
=== Body ===
```

Out[1]=
```
{{"name":"Person","superClass":"","superClasses":[],"alias":null,"abstract":false,"strictmode":false,"
    clusters":[12],"defaultCluster":12,"clusterSelection":"round-robin","records":0,"properties":[{"
    name":"personDOB","type":"DATE","mandatory":true,"readonly":false,"notNull":true,"min":null,"max"
    :null,"regexp":null,"collate":"default"},{"name":"personName","type":"STRING","mandatory":true,"
    readonly":false,"notNull":true,"min":null,"max":null,"regexp":null,"collate":"ci"},{"name":"
    rentCar","linkedClass":"Car","type":"LINK","mandatory":false,"readonly":false,"notNull":false,"
    min":null,"max":null,"regexp":null,"collate":"default"}]}, 200}
```

The postfix operation of **ODBgetDataset** transforms JSON Output above into a Wolfram Dataset Further processing can be applied on the resulting Dataset.

In[2]:=
```
classInfoDS = classInfo // ODBgetDataset
```

Out[2]=

name	Person
superClass	
superClasses	{}
alias	Null
abstract	False
strictmode	False
clusters	{12}
defaultCluster	12
clusterSelection	round-robin
records	0
properties	{<| name → personDOB, linkedClass → KeyAbsent, type → DATE, ⋯6 |>, ⋯2}

3 levels | 11 rows

With the package function **ODBgetFieldAttributes** we can view specific attributes for all properties (fields) of the Car class

In[3]:=
```
ODBgetFieldAttributes[classInfoDS, {"name", "linkedClass", "type", "mandatory", "notNull", "collate"}]
```

Out[3]=

name	linkedClass	type	mandatory	notNull	collate
personDOB	KeyAbsent	DATE	True	True	default
personName	KeyAbsent	STRING	True	True	ci
rentCar	Car	LINK	False	False	default

2 levels | 3 rows

Get Record (1)

Get Record - Returns a JSON structured document with data and metadata fields that represents an OrientDB record.

In[1]:=
```
ODBapi[com → "getRecord", db → "DemoDB", class → "Car", id → "11:0", debug → True]
```

```
http://localhost:2480/document/DemoDB/11:0
=== Body ===
```

Out[1]=
```
{{"@type":"d","@rid":"#11:0","@version":1,"@class":"Car","brand":"FIAT","model":"Punto","year":"2000-
   01-01"}, 200}
```

In[2]:=
```
% // ODBgetDataset
```

Out[2]=

@type	d
@rid	#11:0
@version	1
@class	Car
brand	FIAT
model	Punto
year	2000-01-01

1 level | 7 elements

Get OSQL (1)

Executes an OrientDB SQL query against the database. The SQL command is read-only. It is executed via the GET method of HTTP and therefore it cannot change the database.

```
In[1]:= ODBapi[com → "getOSQLCommand", db → "DemoDB", sql → "select from Car where brand='FIAT'", debug -> True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
select from Car where brand='FIAT'
```

```
Out[1]= {{"result":[{"@type":"d","@rid":"#11:0","@version":1,"@class":"Car","brand":"FIAT","model":"Punto","
    year":"2000-01-01"}]}, 200}
```

Update Commands (5)

Update Database (1)

Update Database command is executed via SQL - ALTER DATABASE operation. This command updates database settings.

Change DATETIMEFORMAT attribute in the database to use ISO 8601 dates

```
In[1]:= ODBapi[com → "updDatabase", db → "DemoDB", attribnam → "DATETIMEFORMAT",
 attribval → "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
ALTER DATABASE DATETIMEFORMAT yyyy-MM-dd'T'HH:mm:ss.SSS'Z'
```

```
Out[1]= {, 204}
```

Enable Lightweight Edges, i.e. bidirectional links

```
In[2]:= ODBapi[com → "updDatabase", db → "DemoDB",
 attribnam → "custom useLightweightEdges=", attribval → "true", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
ALTER DATABASE custom useLightweightEdges= true
```

```
Out[2]= {, 204}
```

Update Class (1)

Update Class command is executed via SQL - ALTER CLASS operation. This command alters a class in the schema.

```
In[1]:= ODBapi[com → "updClass", db → "DemoDB", class → "Person",
 attribnam → "SHORTNAME", attribval → "P", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
ALTER CLASS Person SHORTNAME P
```

```
Out[1]= {, 204}
```

Update Property (1)

Update Property command is executed via SQL - ALTER PROPERTY operation. This command alter's a class's property in the schema.

```
In[1]:= ODBapi[com → "updProperty", db → "DemoDB", class → "Person",
         propnam → "personName", attribnam → "MANDATORY", attribval → "false", debug → True]
Out[1]= {, 204}
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
ALTER PROPERTY Person.personName MANDATORY false
```

```
Out[1]= {, 204}
```

Update Property Value(s) (1)

Update property values for all records in a class that have that property (field) or insert key-value pairs in the Document record if that field does not exist.

```
In[1]:= ODBapi[com → "updPropertyValues", db → "DemoDB",
         class → "Car", propnam → "cc", propval → "1200", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
UPDATE Car set cc=1200
```

```
Out[1]= {{"result":[{"@type":"d","@version":0,"value":2}]}, 200}
```

Update the property value of a single record

```
In[2]:= ODBapi[com → "updPropertyValue", db → "DemoDB", class → "Car",
         id → "11:0", propnam → "year", propval → "\"2001-01-01\"", debug → True]
```

```
http://localhost:2480/command/DemoDB/sql
=== Body ===
UPDATE 11:0 set year="2001-01-01"
```

```
Out[2]= {{"result":[{"@type":"d","@version":0,"value":1}]}, 200}
```

Update Record (1)

Update a record via the PUT-Document HTTP API command. With the default OrientDB update mode the entire document is replaced.

First we modify the Rule expression of the **addDOCUMENT example** and convert it to a JSON string using built-in **ToAssociations** function and **DBexpressionToJSON** functions of the **Utilities Package.**

```
In[1]:= johnJSON = ReplacePart[john // ToAssociations, {
          {"telephones", "business"} → "2108880809",
          {"telephones", "mobile"} → "6972020202"}] // DBexpressionToJSON;
```

Then we pass the result as the value of record → option

```
In[2]:= ODBapi[com → "updRecordPUT", db → "DemoDB",
         class → "Person", id → "12:0", record → johnJSON, debug → True]
```

```
Out[2]= {{"@type":"d","@rid":"#12:0","@version":7,"@class":"Person","telephones":{"home":"2104566345","
    business":"2108880809","mobile":"6972020202"},"firstName":"John","lastName":"Brown","DOB":"1971-
    10-01","age":44}, 200}
```

```
http://localhost:2480/document/DemoDB/12:0
=== Body ===
{
    "telephones": {
        "home": "2104566345",
        "business": "2108880809",
        "mobile": "6972020202"
    },
    "firstName": "John",
    "lastName": "Brown",
    "DOB": "1971-10-01",
    "age": 44
}
```

Update a record via the PATCH-Document HTTP API command. This will update the record Document with only the difference to apply. The ver → here is mandatory.

Build an expression with the telephones to change only

```
In[3]:= telcom = <|"telephones" →
          Thread[{"home", "business", "mobile", "pager"} →
            {"2104566345", "2109990909", "6973030303", "444"}]|> // DBexpressionToJSON;
```

```
In[4]:= ODBapi[com → "updRecordPATCH", db → "DemoDB",
         class → "Person", id → "12:0", record → telcom, ver → 8, debug → True]
```

```
http://localhost:2480/document/DemoDB/12:0
=== Body ===
{"@class":"Person", "@version":8,
    "telephones": {
        "home": "2104566345",
        "business": "2109990909",
        "mobile": "6973030303",
        "pager": "444"
    }
}
```

Import/Export (2)

Import (1)

Export (1)

Login/Logout (2)

Login (1)

Connect to a remote server using basic authentication via GET-Connect HTTP method

```
In[1]:= ODBapi[com → "login", db → "GratefulDeadConcerts", usr → "admin", pwd -> "admin", debug → True]
```

```
http://localhost:2480/connect/GratefulDeadConcerts
=== Body ===
```

```
Out[1]= {, 204}
```

Logout (1)

Disconnect from server via GET-Disconnect HTTP method

```
In[1]:= ODBapi[com -> "logout", debug → True]
```

```
http://localhost:2480/disconnect
=== Body ===
```

```
Out[1]= $Failed
```

MORE ABOUT

- OrientDB Package

RELATED LINKS

- ODBgetDataset . ODBgetFieldAttributes