



Denodo OData2 Custom Wrapper - User Manual

Revision 20160922

NOTE

This document is confidential and proprietary of **Denodo Technologies**.
No part of this document may be reproduced in any form by any means without prior written authorization of **Denodo Technologies**.

Copyright © 2016
Denodo Technologies Proprietary and Confidential

CONTENTS

| | |
|---|-----------|
| 1 INTRODUCTION..... | 3 |
| 2 ODATA SERVICES..... | 4 |
| 2.1 DENODO ODATA2 CUSTOM WRAPPER..... | 4 |
| 2.2 OTHER WAYS TO CONSUME ODATA SOURCES..... | 8 |
| 3 REFERENCES..... | 13 |

1 INTRODUCTION

OData is a protocol to access data created by Microsoft. It provides CRUD operations and is similar to JDBC or ODBC but not limited to databases.

OData uses protocols ATOM and JSON and the requests use a REST model. For this reason, OData is an implementation of the RESTful API that describes the data and their model.

2 ODATA SERVICES

There are several ways to access to an OData service from Virtual DataPort:

2.1 DENODO ODATA2 CUSTOM WRAPPER

The Denodo OData2 Custom Wrapper allows you to access OData services even if they require authentication (HTTP BASIC or NTLM supported) or behind a proxy server (which might also be authenticated).

2.1.1 Import the Custom Wrapper

To import the custom wrapper, follow these steps:

1. In the VDP Administration Tool, go to: File → Extensions → Jar management
2. Click on “Create” button and select the “denodo-odata2-wrapper-5.0-{version}-jar-with-dependencies.jar” file downloaded from Denodo SupportSite.

2.1.2 Create the OData data source

To create a new OData custom data source:

1. In the VDP Administration Tool, go to: File → New... → Data source → Custom
2. In the “Create a New Custom Data Source” window, do the following:
 - Set a name for the new OData data source in the “Name” field.
 - Click on “Select Jars” and select the file imported in the previous section.
 - The “Class name” field must be filled with:
`com.denodo.connect.odata.wrapper.ODataWrapper`
3. Click on “Ok” button.

2.1.3 Create the base view

To create a new base view using the OData data source:

1. Double-click on the OData data source and then click on “Create base view”.
2. Set the parameters as follows:
 - **Service Endpoint** (*mandatory*): is the URL to the OData service. Must be something like: `http://services.odata.org/OData/OData.svc/`
 - **Entity Collection** (*mandatory*): must be one of the collections defined into the OData service.
 - **Service Format** (*mandatory*): is the format used by the OData custom wrapper to access to the OData service. Must be one of these:
 - i. JSON
 - ii. XML-Atom
 - **Service Version**: if specified, the custom wrapper try to force the compatibility of the OData service with one of these versions:
 - i. V1
 - ii. V2
 - **Expand Related Entities**: if checked, the references to other entities appear directly in the main entity as arrays or registers.
 - **Enable Pagination**: if checked, two parameters are added to the view to permit the pagination of the results:
 - i. `fetch_size`
 - ii. `offset_size`
 - **Use NTLM Authentication**: if checked, the fields “User” and “Password” will use NTLM Authentication instead of HTTP Basic Authentication. A NTLM domain could be used using the field “NTLM Domain”.
 - **User**: OData service user for HTTP Basic Authentication or NTLM Authentication, this is an optional parameter.
 - **Password**: OData service password for HTTP Basic Authentication or NTLM Authentication, this is an optional parameter.
 - **Proxy Host**: host to connect to the client through a proxy, this is an optional parameter.

- **Proxy Port:** port to connect to the client through a proxy, this is an optional parameter.
- **Proxy User:** user for the authentication proxy, this is an optional parameter.
- **Proxy Password:** password for the authentication proxy, this is an optional parameter.
- **NTLM Domain:** OData service domain for NTLM Authentication, this is an optional parameter.
- **Timeout.** Maximum time (in milliseconds) the custom wrapper will wait for a query to finish. If it is empty, it will wait indefinitely until the sentence ends.

2.1.4 Example

1. Create a base view over this test service:
 - a)
 - Service Endpoint = [http://services.odata.org/V2/\(S\(gsrk2wcskjydxw2iixw3kvdr\)\)/OData/OData.svc/](http://services.odata.org/V2/(S(gsrk2wcskjydxw2iixw3kvdr))/OData/OData.svc/)
 - Entity = Products
 - Service Format = XML-Atom
 - Expand Related Entities = false
 - Enable Pagination = false
 - b) This is the same option but accessing with a proxy
 - Service Endpoint = [http://services.odata.org/V2/\(S\(gsrk2wcskjydxw2iixw3kvdr\)\)/OData/OData.svc/](http://services.odata.org/V2/(S(gsrk2wcskjydxw2iixw3kvdr))/OData/OData.svc/)
 - Entity = Products
 - Service Format = XML-Atom
 - Expand Related Entities = false
 - Use NTLM Authentication = false
 - Enable Pagination = false
 - Enable Pagination = false
 - Proxy Host=proxy.denodo.com
 - Proxy Port=3128
 - Proxy User=guest
 - Proxy Password=*****
 - Timeout = 1000000

Edit Wrapper Parameter values

Enter values for the following wrapper parameters:

Service Endpoint *

Entity Collection *

Service Format *

Service Version

☐ Expand Related Entities

☐ Use NTLM Authentication

☐ Enable Pagination

User

Password

Proxy Host

Proxy Port

Proxy User

Proxy Password

NTLM Domain

Timeout

2. The schema of the base view is shown and you can rename it.

View schema **Metadata**

View name:

| <input type="checkbox"/> | PK | Field Name | Field Type | |
|--------------------------|----|------------------|------------|--|
| <input type="checkbox"/> | | id | int | <input type="text" value="v"/> <input type="button" value="edit"/> |
| <input type="checkbox"/> | | name | text | <input type="text" value="v"/> <input type="button" value="edit"/> |
| <input type="checkbox"/> | | description | text | <input type="text" value="v"/> <input type="button" value="edit"/> |
| <input type="checkbox"/> | | releasedate | date | <input type="text" value="v"/> <input type="button" value="edit"/> |
| <input type="checkbox"/> | | discontinueddate | date | <input type="text" value="v"/> <input type="button" value="edit"/> |
| <input type="checkbox"/> | | rating | int | <input type="text" value="v"/> <input type="button" value="edit"/> |
| <input type="checkbox"/> | | price | double | <input type="text" value="v"/> <input type="button" value="edit"/> |

3. After clicking on “Ok”, you can execute queries (SELECT, INSERT, UPDATE or DELETE), for example:

- SELECT * FROM products WHERE id = 6;
- INSERT INTO products (id,name,description,releasedate,rating,price) VALUES

```
(9, 'HDTV', '32 inch 720p television', NOW(), 2, 600);
```

- UPDATE products SET price = 800 WHERE id = 9;
- DELETE FROM products WHERE ID = 9;

2.1.5 Known Limitations

This custom data source currently only works with OData versions 1.0 or 2.0.

OData version 3.0 is partially supported interpreting it as a lower version, but this method may not work. More information:

<http://code.google.com/p/odata4j/wiki/Roadmap>

You can't filter elements specified obtained through "expand" related entities. VDP must post-filter these items using ROW syntax in the query.

The insertion and the update of complex fields are not supported.

The insertion of arrays are not supported.

2.2 OTHER WAYS TO CONSUME ODATA SOURCES

2.2.1 Using ATOM/XML

Is possible to access to OData server through an URL. For example, the following URL returns all the entities of the OData server:

<http://services.odata.org/OData/OData.svc/>

Using this URL you can access to all entities of one type:

<http://services.odata.org/OData/OData.svc/Products>

And you can access to one entity using the identifier:

[http://services.odata.org/OData/OData.svc/Products\(0\)](http://services.odata.org/OData/OData.svc/Products(0))

2.2.1.1 Importing ATOM/XML into VDP

1. In the VDP Administration Tool, go to: File → New... → Data source → XML
2. Set the parameters as follows:
 - a. **Name:** the name of the new XML data source.
 - b. **Data route:** "HTTP Client" configured as:

- i. HTTP method: GET
 - ii. URL: URL to the entities. For example:
<http://services.odata.org/OData/OData.svc/Products>
3. Create a new base view:
 - a. We can only recover data selecting “entry” in the section “Stream output at specified level”.
4. When clicking “Ok” we have a similar base view to the next one:

| View schema | | Metadata |
|---|---|------------|
| View name: <input type="text" value="xml"/> | | |
| <input type="checkbox"/> | Field Name | Field Type |
| <input type="checkbox"/> | xmlbase_0 | text |
| <input type="checkbox"/> | id | text |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> title | xml_title |
| <input type="checkbox"/> | updated | text |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> link | xml_link |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> entry | xml_entry |

5. If we execute the view, the results are located into the field “entry”:

Execute

Quick Query

Specify Where Expression

Execute

Query plan

SELECT * FROM xml CONTEXT ('i18n'='us_pst', 'cache_wait_for_load'='true')

Query Results

Results

Execution Trace

Query: SELECT * FROM xml CONTEXT ('i18n'='us_pst', 'cache_wait_for_load'='true') TRAC

Total rows received: 1 (shown 1)

| xmlbase_0 | id | title | updated | link | entry |
|--------------------------|--------------------------|---------------|----------------------|---------------|------------|
| http://services.odata... | http://services.odata... | [Register]... | 2016-01-20T11:03:... | [Register]... | [Array]... |

2.2.2 Using JSON

To access OData using JSON only is necessary to add the following parameter to the URL:
 ?\$format=JSON

It's also possible to access OData using JSON adding the following parameters to the HTTP header when doing a GET:

Accept: application/json

The following is an example of using the URL to access to OData through JSON:

[http://services.odata.org/OData/OData.svc/Products?\\$format=json](http://services.odata.org/OData/OData.svc/Products?$format=json)

2.2.2.1 Importing JSON into VDP

The steps to access to OData using JSON are:

1. In the VDP Administration Tool, go to: File → New... → Data source → JSON
2. Set the parameters as follows:
 - **Name:** the name of the new data source.
 - **Data route:** "HTTP Client" configured as:
 - HTTP method: GET
 - URL: URL to the entities in JSON format. For example: [http://services.odata.org/OData/OData.svc/Products?\\$format=json](http://services.odata.org/OData/OData.svc/Products?$format=json)
3. Create a base view from the new datasource:
 - We can get only the data setting the JSON root as: /JSONFile/value

| PK | Field Name | Field Type |
|--------------------------|------------------|---------------|
| <input type="checkbox"/> | odatametadata_0 | text |
| <input type="checkbox"/> | Evalue | xmljson_value |
| <input type="checkbox"/> | id | int |
| <input type="checkbox"/> | name | text |
| <input type="checkbox"/> | description | text |
| <input type="checkbox"/> | releasedate | text |
| <input type="checkbox"/> | rating | int |
| <input type="checkbox"/> | price | double |
| <input type="checkbox"/> | discontinueddate | text |
| <input type="checkbox"/> | odatatype_0 | text |

4. When clicking "Ok" we have a similar base view to the next one:

5. Data are located in the field “array_value”:

Execute X

Quick Query Specify Where Expression

Execute Query plan

Current sentence:

```
SELECT * FROM xmljson CONTEXT ('i18n='us_pst', 'cache_wait_for_load'=true')
```

☐ Do not use cache ☐ Invalidate existing results ☒ Limit rows 150 ☒ Execute with TRACE

Query Results X

Results Execution Trace

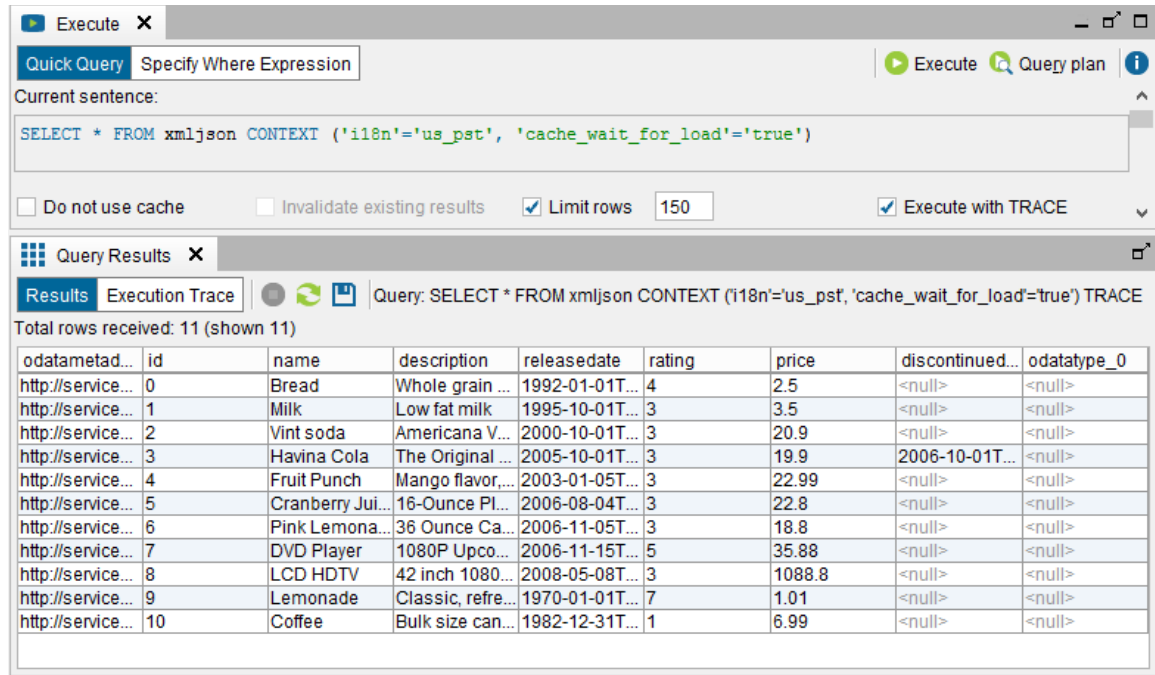
Query: SELECT * FROM xmljson CONTEXT ('i18n='us_pst', 'cache_wait_for_load'=true') TRACE

Total rows received: 1 (shown 1)

RESU... -> value

| id | name | description | releasedate | rating | price | discontinuedd... | odatatype_0 |
|----|-----------------|--------------------|-----------------|--------|--------|------------------|----------------|
| 0 | Bread | Whole grain br... | 1992-01-01T0... | 4 | 2.5 | <null> | <null> |
| 1 | Milk | Low fat milk | 1995-10-01T0... | 3 | 3.5 | <null> | <null> |
| 2 | Vint soda | Americana Vari... | 2000-10-01T0... | 3 | 20.9 | <null> | <null> |
| 3 | Havina Cola | The Original K... | 2005-10-01T0... | 3 | 19.9 | 2006-10-01T0... | <null> |
| 4 | Fruit Punch | Mango flavor, 8... | 2003-01-05T0... | 3 | 22.99 | <null> | <null> |
| 5 | Cranberry Juice | 16-Ounce Plas... | 2006-08-04T0... | 3 | 22.8 | <null> | <null> |
| 6 | Pink Lemonade | 36 Ounce Can... | 2006-11-05T0... | 3 | 18.8 | <null> | <null> |
| 7 | DVD Player | 1080P Upconv... | 2006-11-15T0... | 5 | 35.88 | <null> | <null> |
| 8 | LCD HDTV | 42 inch 1080p ... | 2008-05-08T0... | 3 | 1088.8 | <null> | <null> |
| 9 | Lemonade | Classic, refres... | 1970-01-01T0... | 7 | 1.01 | <null> | ODataDemo.F... |
| 10 | Coffee | Bulk size can o... | 1982-12-31T0... | 1 | 6.99 | <null> | ODataDemo.F... |

6. Data can be directly accessed if you specified the root “/JSONFile/value”:



The screenshot shows the Denodo Query Editor interface. The 'Execute' tab is active, displaying the query: `SELECT * FROM xmljson CONTEXT ('i18n'='us_pst', 'cache_wait_for_load'='true')`. Below the query, there are checkboxes for 'Do not use cache', 'Invalidate existing results', 'Limit rows' (set to 150), and 'Execute with TRACE' (checked). The 'Query Results' tab is also visible, showing the execution trace and the results of the query. The results are displayed in a table with 11 rows and 8 columns.

| odatametad... | id | name | description | releasedate | rating | price | discontinued... | odatatype_0 |
|-------------------|----|------------------|-------------------|----------------|--------|--------|-----------------|-------------|
| http://service... | 0 | Bread | Whole grain ... | 1992-01-01T... | 4 | 2.5 | <null> | <null> |
| http://service... | 1 | Milk | Low fat milk | 1995-10-01T... | 3 | 3.5 | <null> | <null> |
| http://service... | 2 | Vint soda | Americana V... | 2000-10-01T... | 3 | 20.9 | <null> | <null> |
| http://service... | 3 | Havina Cola | The Original ... | 2005-10-01T... | 3 | 19.9 | 2006-10-01T... | <null> |
| http://service... | 4 | Fruit Punch | Mango flavor... | 2003-01-05T... | 3 | 22.99 | <null> | <null> |
| http://service... | 5 | Cranberry Jui... | 16-Ounce Pl... | 2006-08-04T... | 3 | 22.8 | <null> | <null> |
| http://service... | 6 | Pink Lemona... | 36 Ounce Ca... | 2006-11-05T... | 3 | 18.8 | <null> | <null> |
| http://service... | 7 | DVD Player | 1080P Upco... | 2006-11-15T... | 5 | 35.88 | <null> | <null> |
| http://service... | 8 | LCD HDTV | 42 inch 1080... | 2008-05-08T... | 3 | 1088.8 | <null> | <null> |
| http://service... | 9 | Lemonade | Classic, refre... | 1970-01-01T... | 7 | 1.01 | <null> | <null> |
| http://service... | 10 | Coffee | Bulk size can... | 1982-12-31T... | 1 | 6.99 | <null> | <null> |

3 REFERENCES

Odata official page

- Documentation:
 - <http://www.odata.org/docs/>
- Libraries:
 - <http://www.odata.org/libraries/>

Wikipedia article:

- http://en.wikipedia.org/wiki/Open_Data_Protocol

OData references into the Microsoft webpage:

- Create and Consume JSON-Formatted OData:
 - <http://msdn.microsoft.com/es-es/magazine/jj190799.aspx>
- Building Rich Internet Apps with the Open Data Protocol:
 - <http://msdn.microsoft.com/es-es/magazine/ff714561.aspx>
- OData Operations:
 - <http://www.odata.org/documentation/odata-v2-documentation/operations/>
- Examples:
 - <http://msdn.microsoft.com/en-us/library/ff478141.aspx>

Web pages with OData examples:

- Example read-only service in the official web site:
 - <http://services.odata.org/OData/OData.svc/>
 - [http://services.odata.org/OData/OData.svc/\\$metadata](http://services.odata.org/OData/OData.svc/$metadata)
 - <http://services.odata.org/OData/OData.svc/Products>
 - [http://services.odata.org/OData/OData.svc/Products\(0\)](http://services.odata.org/OData/OData.svc/Products(0))
 - [http://services.odata.org/OData/OData.svc/Products?\\$format=json](http://services.odata.org/OData/OData.svc/Products?$format=json)
 - [http://services.odata.org/OData/OData.svc/Products\(0\)?\\$format=json](http://services.odata.org/OData/OData.svc/Products(0)?$format=json)
- Example read-write service in the official web site:
 - [http://services.odata.org/V2/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/](http://services.odata.org/V2/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/)
 - [http://services.odata.org/V2/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/Categories](http://services.odata.org/V2/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/Categories)
 - [http://services.odata.org/V2/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/Products](http://services.odata.org/V2/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/Products)
 - [http://services.odata.org/V2/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/Suppliers](http://services.odata.org/V2/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/Suppliers)
- Example OData installing the module odata-server of JayData server

- You can install locally this module to test the Basic Authentication, the instructions in this url: <https://www.npmjs.org/package/odata-server>
- In the tab Live Services there are more examples:
<http://www.odata.org/ecosystem/>