



Denodo OData4 Custom Wrapper - User Manual

Revision 20180927

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 © 2018
Denodo Technologies Proprietary and Confidential

CONTENTS

1 INTRODUCTION.....	3
2 ODATA SERVICES.....	4
2.1 DENODO ODATA4 CUSTOM WRAPPER.....	4
2.2 OTHER WAYS TO CONSUME ODATA SOURCES.....	10
3 REFERENCES.....	14

1 INTRODUCTION

OData is a protocol to access data created by Microsoft. In the version 4 OData has become a standardized protocol of the OASIS OData Technical Committee (TC). 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 ODATA4 CUSTOM WRAPPER

The Denodo OData4 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). In addition, it supports HTTPS connections with a valid certificate.

2.1.1 Import the Custom Wrapper

To import the custom wrapper, follow these steps:

1. In the VDP Administration Tool, go to:
 - Until Denodo 6.0: File → Jar management
 - From Denodo 7.0: File → Extension management
2. Click on “Create” button and select the “denodo-odata4-wrapper-`{denodo-version}-{version}-jar-with-dependencies.jar`” file downloaded from Denodo Support Site.

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 in the OData service, specified as the complete url path (without parameters) required to reach that collection. For example: "Customers" or "Company('DND')/currentCustomers".
- **Entity Name**: optionally, the name of the entity type modeling the data returned by the Entity Collection, as defined in the service's EDM. If no value is specified, then the Entity Collection will be considered to appear in the Service Document and the Entity Name will be inferred from there. For example: "Customers" -> "Customer".
- **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
- **Custom Query Options**: Data service-specific information to be placed in the data service URI. A custom query option is any query option URI with the following form `customOption = customValue`.
In this parameter you can put several query options separated by &. Custom query options MUST NOT begin with a \$ or @ character, according to the the standard of OData 4.

For example: `http://host/service/Products?debug-mode=true`

- **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`
- **Load Streams**: if checked, Stream properties and Stream entities will be loaded as BLOB objects instead of the link that allows the access to the media resources.
- **Pass-through session credentials**: if checked, the value of the login and password fields are used for introspection. During execution, the credentials of the user authenticated in VDP are used.
- **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.
- **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.

- **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.
- **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".
- **Use OAuth2:** if checked, the protocol OAuth 2.0 will be used for authorization. With this option the fields: Access Token, Refresh Token, Client Id, Client Secret, and Token Endpoint Url are mandatory. You can use the Oauth credentials wizard to obtain the Access Token and the Refresh Token. You can read more in the VDP ADMINISTRATION GUIDE in the subsection of Oauth.
- **Access Token:** You can use the Oauth credentials wizard to get it
- **Refresh Token:** You can use the Oauth credentials wizard to get it. It is used when the Access Token has expired to get a new access token.
- **Client Id:** consumer key from the remote access application definition.
- **Client Secret:** consumer secret from the remote access application definition.
- **Token Endpoint URL:** it is URL to make OAuth refresh request when the Access Token has expired.
- **Refr. Token Auth. Method:** controls how the credentials are sent to the service when requesting a new OAuth access token. There are two options:
 - i. Include the client credentials in the body of the request (this is the default option)
 - ii. Send client credentials using the HTTP Basic authentication scheme
- **HTTP Headers:** custom headers to be used in the underlying HTTP client, this is an optional parameter. *Note: Multiple HTTP headers are allowed to be added. The format must be as follows:*

field_1="value_1";field_2="value_2";...;field_n="value_n" ;.
Being "field" the name of the header and "value" its value.

2.1.4 Example

1. Create a base view over this test service:

a)

- ☐ Service Endpoint =
[http://services.odata.org/V4/\(S\(gsrk2wcskjdxw2iixw3kvdr\)\)/OData/OData.svc/](http://services.odata.org/V4/(S(gsrk2wcskjdxw2iixw3kvdr))/OData/OData.svc/)
- ☐ Entity Collection = Products
- ☐ Service Format = XML-Atom
- ☐ Expand Related Entities = false
- ☐ Enable Pagination = false
- ☐ Load Streams = false

b) This is the same option but accessing with a proxy

- ☐ Service Endpoint =
[http://services.odata.org/V4/\(S\(gsrk2wcskjdxw2iixw3kvdr\)\)/OData/OData.svc/](http://services.odata.org/V4/(S(gsrk2wcskjdxw2iixw3kvdr))/OData/OData.svc/)
- ☐ Entity Collection = Products
- ☐ Service Format = XML-Atom
- ☐ Expand Related Entities = false
- ☐ Use NTLM Authentication = false
- ☐ Enable Pagination = false
- ☐ Load Streams = 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 *	<input type="text" value="https://services.odata.org/V4/(S(gsrk2wcskjdxdw2iixw3kvdr))/OData/OData.svc/"/>
Entity Collection *	<input type="text" value="Products"/>
Entity Name	<input type="text"/>
Service Format *	<input type="text" value="XML-Atom"/>
Custom Query Options	<input type="text"/>
	<input type="checkbox"/> Expand Related Entities <input type="checkbox"/> Enable Pagination <input type="checkbox"/> Load Streams <input type="checkbox"/> Pass-through session credentials
User	<input type="text"/>
Password	<input type="password"/>
Timeout	<input type="text" value="1000000"/>
Proxy Host	<input type="text" value="proxy.denodo.com"/>
Proxy Port	<input type="text" value="3128"/>
Proxy User	<input type="text" value="guest"/>
Proxy Password	<input type="password" value="•••••"/>
	<input type="checkbox"/> Use NTLM Authentication <input type="checkbox"/> Use OAuth2
NTLM Domain	<input type="text"/>
Access Token	<input type="text"/>
Refresh Token	<input type="text"/>
Client Id	<input type="text"/>
Client Secret	<input type="text"/>
Token Endpoint URL	<input type="text"/>
Refr. Token Auth. Method	<input type="text"/>
HTTP Headers	<input type="text"/>

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

View schema

Metadata

View name: odata

<input type="checkbox"/>	PK	Field Name	Field Type	
<input type="checkbox"/>		id	int	 
<input type="checkbox"/>		name	text	 
<input type="checkbox"/>		description	text	 
<input type="checkbox"/>		releasedate	date	 
<input type="checkbox"/>		discontinueddate	date	 
<input type="checkbox"/>		rating	int	 
<input type="checkbox"/>		price	double	 

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 version 4.0. The previous versions of Odata are not supported by this wrapper. There is other custom wrapper to access to old versions of OData.

Filtering of elements by array properties is not supported. OData does not allow this kind of searches.

Filtering by media read links properties is not supported.

The insertion of arrays is not supported.

The authentication using NTLM through a proxy is not supported.

The insertion or update of media file properties is not supported.

Addressing derived types properties is not available. When you have entities with a type derived from the declared type of the requested collection, their properties will be added to the schema of the base view but you can't project these properties separately.

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/V4/OData/OData.svc/>

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

<http://services.odata.org/V4/OData/OData.svc/Products>

And you can access to one entity using the identifier:

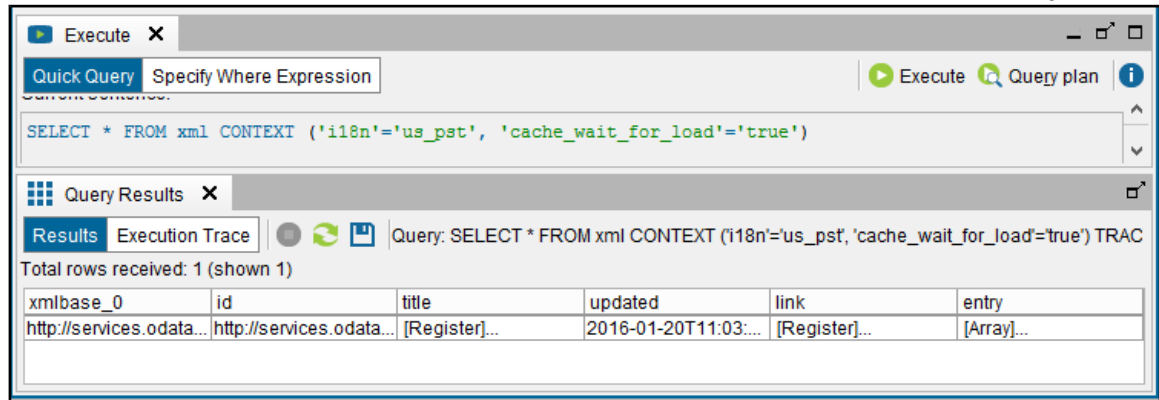
[http://services.odata.org/V4/OData/OData.svc/Products\(0\)](http://services.odata.org/V4/OData/OData.svc/Products(0))

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/V4/OData/OData.svc/Products>
3. Create a new base view:
 - a. We can only recover data selecting "entry" in the section "Stream output level".
4. When clicking "Ok" we have a similar base view to the next one:

View schema		Metadata
View name: xml		
<input type="checkbox"/> PK	Field Name	Field Type
<input type="checkbox"/>	xmlbase_0	text
<input type="checkbox"/>	id	text
<input type="checkbox"/>	⊕title	📄 xml_title
<input type="checkbox"/>	updated	text
<input type="checkbox"/>	⊕link	📄 xml_link
<input type="checkbox"/>	⊕entry	📄 xml_entry

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



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/V4/OData/OData.svc/Products?\\$format=json](http://services.odata.org/V4/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/V4/OData/OData.svc/Products?\\$format=json](http://services.odata.org/V4/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`
4. When clicking “Ok” we have a similar base view to the next one:

View schema Metadata		
View name: xmljson		
<input type="checkbox"/> PK	Field Name	Field Type
<input type="checkbox"/>	odatametadate_0	text
<input type="checkbox"/>	value	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

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

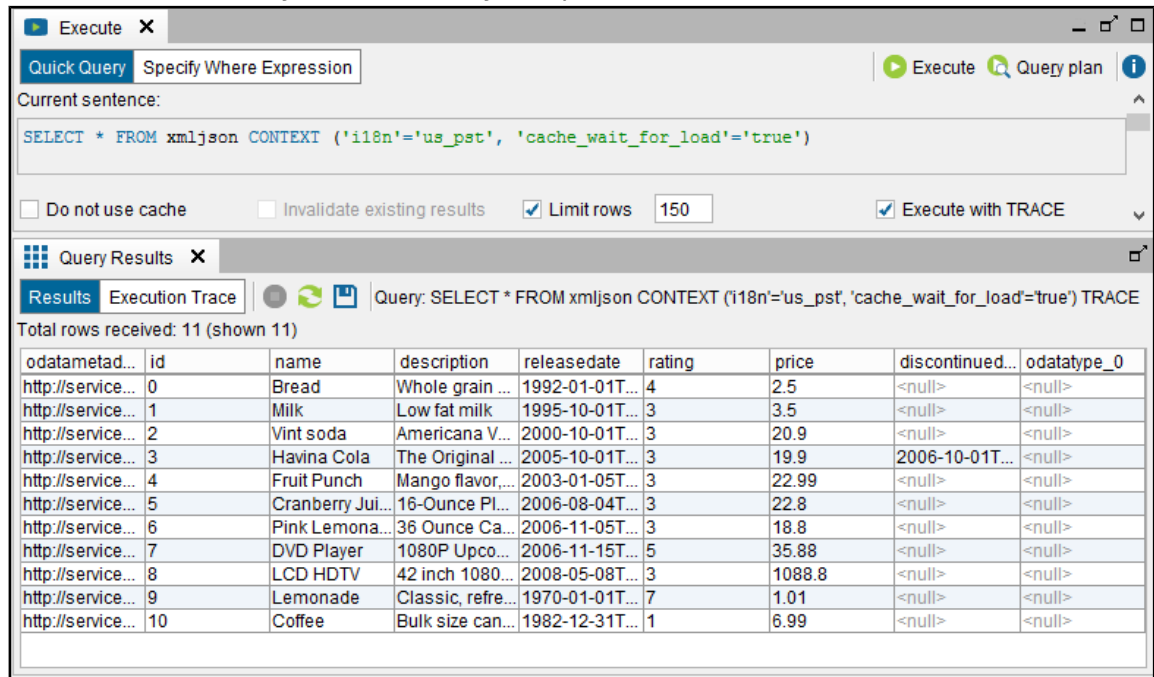
Query: SELECT * FROM xmljson CONTEXT ('i18n'='us_pst', 'cache_wait_for_load'='true') TRACE

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 Execute interface. At the top, there's a "Quick Query" section with a "Specify Where Expression" button. Below it, the "Current sentence:" field contains the query: `SELECT * FROM xmljson CONTEXT ('i18n'='us_pst', 'cache_wait_for_load'='true')`. To the right of the query field are buttons for "Execute", "Query plan", and an information icon. Below the query field, there are checkboxes for "Do not use cache", "Invalidate existing results", "Limit rows" (set to 150), and "Execute with TRACE".

The "Query Results" section is active, showing a table with 11 rows. The table has columns: `odatametad...`, `id`, `name`, `description`, `releasedate`, `rating`, `price`, `discontinued...`, and `odatatype_0`. The data rows show various products like Bread, Milk, Vint soda, Havina Cola, Fruit Punch, Cranberry Jui..., Pink Lemona..., DVD Player, LCD HDTV, Lemonade, and Coffee.

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/V4/OData/OData.svc/>
 - [http://services.odata.org/V4/OData/OData.svc/\\$metadata](http://services.odata.org/V4/OData/OData.svc/$metadata)
 - <http://services.odata.org/V4/OData/OData.svc/Products>
 - [http://services.odata.org/V4/OData/OData.svc/Products\(0\)](http://services.odata.org/V4/OData/OData.svc/Products(0))
 - [http://services.odata.org/V4/OData/OData.svc/Products?\\$format=json](http://services.odata.org/V4/OData/OData.svc/Products?$format=json)
 - [http://services.odata.org/V4/OData/OData.svc/Products\(0\)?\\$format=json](http://services.odata.org/V4/OData/OData.svc/Products(0)?$format=json)
- Example read-write service in the official web site:
 - [http://services.odata.org/V4/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/](http://services.odata.org/V4/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/)
 - [http://services.odata.org/V4/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/Categories](http://services.odata.org/V4/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/Categories)
 - [http://services.odata.org/V4/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/Products](http://services.odata.org/V4/(S(gsrk2wckjydxw2iixw3kvdr))/OData/OData.svc/Products)
 - [http://services.odata.org/V4/\(S\(gsrk2wckjydxw2iixw3kvdr\)\)/OData/OData.svc/Suppliers](http://services.odata.org/V4/(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/>