Denodo OData 2.0 Service - User Manual

Revision 20170518

CONTENTS

1 OVERVIEW

[OData](#) (**Open Data Protocol**) is an OASIS standard. It is an open protocol for building and consuming RESTful APIs. It means that OData enables the creation of HTTP-based data services, which allow resources identified using Uniform Resource Identifiers (URIs) and defined in a data model, to be published and edited by Web clients using simple HTTP messages.

Denodo OData Service allows Denodo users to connect to the Denodo Platform and query its databases using an **OData 2.0** interface.

More on the OData standard here: http://www.odata.org

Note that this document will use "<VERSION>" as a placeholder for the specific version of the Denodo Platform you are installing the Denodo OData Service for. So this placeholder should be replaced by "5.5", "6.0", etc. depending on your Denodo version.

2 INSTALLATION

The Denodo OData Service distribution consists of:

- A war file (`denodo-odata2-service-<VERSION>.war`)
- A documentation folder containing this user manual (`/doc` folder)
- A scripts folder containing the scripts to install the service into tomcat embedded (`/scripts` folder)

For running the Denodo OData Service you need to deploy the war file in a Java web application container. **Apache Tomcat 7+** (using Java 7 or later) is recommended. For this you have to include the denodo-vdp-jdbcdriver of your platform in the web container, in the case of Apache Tomcat 7+ in the folder `/lib`. You can find the .jar in the folder `$DENODO_HOME\tools\client-drivers\jdbc` or in the folder `$DENODO_HOME\lib\vdp-jdbcdriver-core` if you use Denodo Platform 5.5 or previous versions.

The web application container must provide the data source configuration in order to connect to the Denodo Virtual DataPort server using JNDI. The JNDI name of the resource must be `jdbc/VDPdatabase`. See the **Configuring JNDI resources in Apache Tomcat** for more information.

The Denodo OData service has some properties that can be configured in the web container and the Denodo OData service will fetch their value via JNDI (see the **Configuring JNDI resources in Apache Tomcat section** for more information):

- `odataserver.address` service name, `/denodo-odata.svc` by default.
- `odataserver.serviceRoot` root URI of the service. It is an optional property, empty by default. It should be configured when the OData service is going to be accessed through a **gateway**, so links within the OData response use this URI as root.
  For example, being:
  `odataserver.serviceRoot=https://gw.denodo.com:9000/ODATA/`
  and accessing:
  `http://server001:9090/denodo-odata2-service-<VERSION>/denodo-odata.svc/movies`
  will return this kind of URI within the response:
  `https://gw.denodo.com:9000/ODATA/denodo-odata2-service-<VERSION>/denodo-odata.svc/movies`
- `server.pageSize` default number of returned entries per request (see **Pagination section** for more information)
- `enable.adminUser` boolean value that allows you to deny access to the OData service using the `admin` user if the specified value is `false`.

Besides, the war file has a configuration file in the `WEB-INF/classes` folder called `configuration.properties` that has a default value for the properties explained above:

```
odataserver.address=/denodo-odata.svc
odataserver.serviceRoot=
server.pageSize=1000
enable.adminUser=true
```

The property values specified at the web container as JNDI entries have a higher precedence; therefore if a property's value is established via JNDI the Denodo OData service will disregard the value for this property at the `configuration.properties`.

Once you deployed the war you can use the Denodo OData Service from a web client, using HTTP Basic Authentication with VDP-valid credentials). Therefore you may use URLs that are of the form:

http://localhost:8080/denodo-odata.svc/<DBNAME>

(Note that for the sake of simplicity in this document, we will consider the OData server to be installed at the ROOT context of the web server)

Deploying into the Denodo embedded web container

You can also deploy the service in the internal web container of the Denodo Platform. For this, you have two options, with the option A , you can execute manually the scripts. And with the option B, you can run the start and the stop of the application as a Service in Windows NT/2000/2003/XP/Vista.

2.1.1 Option A

You should follow these steps:

1. Copy the `denodo-odata2-service-<VERSION>.war` into the following location:
   `<DENODO_HOME>/resources/apache-tomcat/webapps/`

2. Create a context xml file for the Denodo Odata Service. You can use the xml template `$DENODO_ODATA_SERVICE_HOME/scripts/denodo-odata2-service-<VERSION>.xml`.

3. Copy the `denodo-odata2-service-<VERSION>.xml` file into the following folder:

```
<DENODO_HOME>/resources/apache-tomcat/conf/DenodoPlatform-
<VERSION>/localhost
```

4. Check that the common.loader property (shared.loader in Denodo 5.5) in the <DENODO_HOME>/resources/apache-tomcat/conf/catalina.properties file includes a reference to the VDP JDBC Driver. If missing, add it with: ${catalina.base}/../../tools/client-drivers/jdbc/denodo-vdp-jdbcdriver-basic.jar or ${catalina.base}/../../lib/vdp-jdbcdriver-core/denodo-vdp-jdbcdriver-basic.jar if you are using Denodo Platform 5.5 or previous versions, separated from other entries with a comma.

5. Create launch scripts for the Denodo Odata Service. You can use the templates odata_service_startup.bat or odata_service_shutdown.bat for Windows or the templates odata_service_startup.sh odata_service_shutdown.sh for UNIX/Linux located at $DENODO_ODATA_SERVICE_HOME/scripts/.

   Make sure to modify the DENODO_HOME variable in the script templates to point to your Denodo installation.

6. Copy the launch scripts into <DENODO_HOME>/bin.

7. Provide the data source configuration in order to connect to the Denodo Virtual DataPort server and specified the property values in those that you want to set a different value to the default (see the **Configuring JNDI resources in Apache Tomcat section** for more information).

8. After copying all the necessary files into the correct directories and configuring the data source and the properties that you want to change its default value specified in the configuration.properties, run the <DENODO_HOME>/bin/odata_service_startup.bat(.sh) launch script and then navigate to:

   ```
   http://localhost:9090/denodo-odata2-service-<VERSION>/denodo-odata.svc/<DBNAME>
   ```

2.1.2 Option B

This option is only for Windows Systems.

1-4. Same as option A

5. Create service script for the Denodo Odata Service(odata2_service_service). You can use the attached template. Make sure to modify the DENODO_HOME variable in the script templates to point to your Denodo installation.

6. Create the file service.conf and copy in the folder <DENODO_HOME>/conf/denodo-odata2-service-service/, you can use the attached template. Make sure to modify the DENODO_HOME variable in the script templates to point to your Denodo installation.

7. Install as a Windows service. Execute "odata2-service-service.bat install"

8. You can manage the installed service from the application of Windows: Services.

After launching of the service you can navigate to:

http://localhost:9090/denodo-odata2-service-<VERSION>/denodo-odata.svc/<DBNAME>

2.2   CONFIGURING JNDI RESOURCES IN APACHE TOMCAT

Data source configuration

2.2.1.1   For Denodo Platform 6.0

1. Declare the JNDI resource in Apache Tomcat's `server.xml` file with the name `jdbc/VDPdatabase`. You must modify the server.xml file at `<DENODO_HOME>/resources/apache-tomcat/conf`. Below, there is an example:

```xml
<GlobalNamingResources>
  <Resource name="jdbc/VDPdatabase"
            auth="Container"
            type="javax.sql.DataSource"
            username="admin" password="admin"
            url="jdbc:vdb://localhost:9999/admin"
            driverClassName="com.denodo.vdp.jdbc.Driver"
            initialSize="5"
            <!-- maxWaitMillis and maxWait for compatibility between
                 Tomcat versions -->
            maxWaitMillis="5000"
            maxWait="5000"
            <!-- maxTotal and maxActive for compatibility between
                 Tomcat versions -->
            maxTotal="20"
            maxActive="20"
            maxIdle="5"
            validationQuery="select 1"
            poolPreparedStatements="true"/>
</GlobalNamingResources>
```

2. Reference the JNDI resource from Apache Tomcat's `context.xml` file. If you deploy the Denodo OData service into the Denodo embedded web container the `context.xml` file is in the `<DENODO_HOME>/resources/apache-tomcat/conf` folder.

```
<Context>
 <ResourceLink name="jdbc/VDPdatabase"
                global="jdbc/VDPdatabase"
                type="javax.sql.DataSource"/>
</Context>
```

<u>For Denodo Platform 5.5</u>

1. Declare the JNDI resource in Apache Tomcat's context.xml file with the name jdbc/VDPdatabase. You must modify the context.xml file at <DENODO_HOME>/resources/apache-tomcat/conf. Below, there is an example:

```
<Context>
 <Resource name="jdbc/VDPdatabase"
          auth="Container"
          type="javax.sql.DataSource"
          username="admin" password="admin"
          url="jdbc:vdb://localhost:9999/admin"
          driverClassName="com.denodo.vdp.jdbc.Driver"
          initialSize="5"
          <!-- maxWaitMillis and maxWait for compatibility between
               Tomcat versions -->
          maxWaitMillis="5000"
          maxWait="5000"
          <!-- maxTotal and maxActive for compatibility between
               Tomcat versions -->
          maxTotal="20"
          maxActive="20"
          maxIdle="5"
          minIdle="0"
          validationQuery="select 1"
          poolPreparedStatements="true"/>

</Context>
```

* The settings described for Denodo Platform 6.0 can works with 5.5 but in the initialization of Tomcat embedded would appear a spurious error that you could ignore.

*The parameter minIdle could be greater than 0. In this way the performance of the connections would be better, but other applications would have maintain open connections that never will be used.

Properties configuration

Using the Apache Tomcat web container you should establish the properties value in the `context.xml` file:

```
<Context>
  <Environment type="java.lang.String" name="odataserver.address"
              value="/denodo-odata.svc"/>
  <Environment type="java.lang.String" name="odataserver.serviceRoot"
              value=""/>  <!-- OPTIONAL PARAMETER -->
  <Environment type="java.lang.Integer" name="server.pageSize"
              value="1000"/>
  <Environment type="java.lang.Boolean" name="enable.adminUser"
              value="false"/>
</Context>
```

If you deploy the service in the internal web container of the Denodo Platform, this file is at `<DENODO_HOME>/resources/apache-tomcat/conf`.

3 FEATURES

Denodo OData Service provides the following main features:

- **Read-only** access to Denodo databases via **OData 2.0**.
- Show metadata of the Denodo database to which the connection is made.
- Address collections.
- Address entries.
- Address properties of an entry.
- Address property values.
- Address links between entries.
- Format results both in AtomPub and/or in JSON (`$format`)
- Query string options
  - `$filter`
  - `$select`
  - `$orderby`
  - `$expand`
  - `$skip`
  - `$top`
  - `$inlinecount`
- Pagination

4 SERVING METADATA

There are two types of metadata documents:

- The **Service Document** that lists all the top-level feeds, which are collections of typed Entries and it is available at the Service Root URI, the root of the OData Service, specifying the database name where we are going to get information, denodo-odata.svc/<DBNAME>. Below, there is an example where the accessible collections of movies database are actor, address, city, country, film and film_actor.

  http://localhost:8080/denodo-odata.svc/movies

  ```
  <?xml version='1.0' encoding='utf-8'?>
  <service              xml:base="http://localhost:8080/denodo-odata.svc/movies/"
  xmlns="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom">
    <workspace>
    <atom:title>Default</atom:title>
    <collection href="actor">
      <atom:title>actor</atom:title>
    </collection>
    <collection href="address">
      <atom:title>address</atom:title>
    </collection>
    <collection href="city">
      <atom:title>city</atom:title>
    </collection>
    <collection href="country">
      <atom:title>country</atom:title>
    </collection>
    <collection href="film">
      <atom:title>film</atom:title>
    </collection>
    <collection href="film_actor">
      <atom:title>film_actor</atom:title>
     </collection>
   </workspace>
  </service>
  ```

- The **Service Metadata Document**, also called **Entity Data Model (EDM)**, that describes the data model exposed in **CSDL** (Common Schema Definition Language), an application of XML. This Entity Data Model is available at the .../$metadata URL and includes five types of structures:

  - Entity Types (also "feeds" or "collections")
  - Associations
  - Entity Sets
  - Association Sets
  - Imported Functions

  http://localhost:8080/denodo-odata.svc/movies/$metadata

  Example of feed:

  ```
  <EntityType Name="actor">
    <Key>
      <PropertyRef Name="actor_id"/>
    </Key>
    <Property Name="actor_id" Type="Edm.Int16" Nullable="false"/>
     <Property Name="first_name" Type="Edm.String" Nullable="true" MaxLength="45"/>
     <Property Name="last_name" Type="Edm.String" Nullable="true" MaxLength="45"/>
     <Property    Name="last_update"    Type="Edm.DateTimeOffset"    Nullable="true"
     Precision="19"/>
  </EntityType>
  ```

Example of association that represents a relationship between `country` and `city`. Every `country` element is related with zero or more `city` elements:

```
<Association Name="country_city">
  <Documentation>
    <Summary>Association between cities and countries</Summary>
  </Documentation>
  <End Type="com.denodo.odata2.country" Multiplicity="1" Role="country"/>
  <End Type="com.denodo.odata2.city" Multiplicity="*" Role="cities"/>
  <ReferentialConstraint>
    <Principal Role="country">
      <PropertyRef Name="country.country_id"/>
    </Principal>
    <Dependent Role="cities">
      <PropertyRef Name="city.country_id"/>
    </Dependent>
  </ReferentialConstraint>
</Association>
```

The Denodo OData Service maps these OData structures to VDP concepts like this:

Denodo ODATA Service	VDP
Entity Type	**View Definition**
Entity Type > Property	*View Column*
Entity Type > Navigation Property	*Association Role*
Association	**Association Definition**
Entity Set	**View Data**
Association Set	**(Associated data)**
Imported Functions	**-**

5 QUERYING DATA: THE BASICS

5.1 QUERYING COLLECTIONS

In the Service Metadata Document (see the **Metadata section** for more information) you can see the entity set names and to see its data you must use the following URL:

```
/denodo-odata.svc/<DBNAME>/collectionName
```

Example (note that, for the sake of simplicity, we are removing the server and port from the example):

```
/denodo-odata.svc/movies/actor
```

5.2 OBTAINING ENTRIES BY PRIMARY KEY

Each entry could be identified using its primary key property:

```
denodo-odata.svc/<DBNAME>/collectionName(keyvalue)
```

Examples:

```
/denodo-odata.svc/movies/actor(1)
/denodo-odata.svc/movies/store_category('F0')
```

The PK can be a compound key, and in this case you must include all values separated by commas:

```
/denodo-odata.svc/<DBNAME>/collectionName(key1,key2)
```

Example:

```
/denodo-odata.svc/movies/film_actor(actor_id=1,film_id=1)
```

Note that, when there is not a defined primary key, this option is unavailable.

5.3 ACCESSING INDIVIDUAL PROPERTIES

Properties of an entry can be accessed individually:

```
/denodo-odata.svc/<DBNAME>/collectionName(key)/propertyName
```

Example:

```
/denodo-odata.svc/movies/actor(1)/first_name
```

Response:

```
{
  "d": {
      "first_name": "PENELOPE"
  }
}
```

5.4   ACCESSING INDIVIDUAL PROPERTY VALUES

The value of a property is available as a raw value:

/denodo-odata.svc/<DBNAME>/collectionName(key)/propertyName/$value

Example:

/denodo-odata.svc/movies/actor(1)/first_name/$value

Response:

PENELOPE

5.5   ACCESSING COMPLEX PROPERTIES

Properties can be complex but they are also accessible. You must point out the property path, from the complex to the simple one:

/denodo-odata.svc/<DBNAME>/collectionName(pk)/propName/complexProp/propName

For example, for the following film_data complex field in a struct_table_film entity:

```
"table_id": "1",
"film_data": {
  "__metadata": {
      "type": "com.denodo.odata2.struct_table_film_film_data"
  },
  "id": "1",
  "title": "ACADEMY DINOSAUR",
  "description": "ELIZABETH"
}
```

...we could perform the following call:

/denodo-odata.svc/movies/struct_table_film(1)/film_data/title

Response:

```
{
  "d": {
      "title": "ACADEMY DINOSAUR"
  }
}
```

5.6 COUNTING ELEMENTS IN A COLLECTION: $COUNT

If you want to know the number of elements (entries) in a collection you have to add $count to the URL:

```
/denodo-odata.svc/<DBNAME>/collectionName/$count
```

Example:

```
/denodo-odata.svc/movies/actor/$count
```

Response:

```
200
```

5.7 ESTABLISHING RESPONSE FORMAT: ATOMPUB VS JSON

OData resources can be represented in AtomPub (XML) or JSON. Using the system query option $format you might specify the format to be used by the OData service to display the response.

The example URLs below will show the same data but represented using the AtomPub format as defined in OData:Atom in the first one and in the second using the JSON format as defined in OData:JSON.

```
/denodo-odata.svc/movies/actor?$format=atom
```
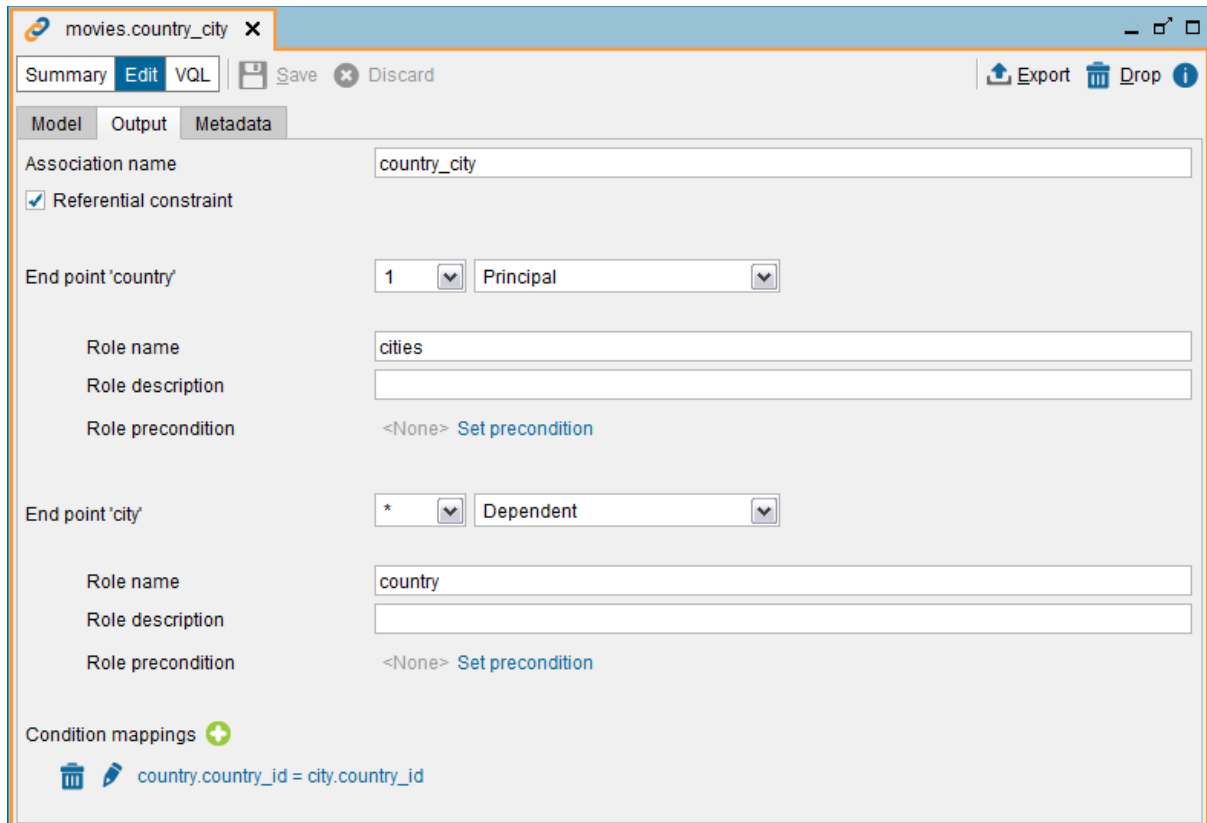
```
/denodo-odata.svc/movies/actor?$format=json
```

An alternative and equally valid way to request a JSON response from the OData Service (note that AtomPub is the default) is by means of specifying an Accept HTTP header in the request, such as:
Accept: application/json

6 NAVIGATING ASSOCIATIONS

6.1 PUBLISHING VDP ASSOCIATIONS THROUGH ODATA

Denodo Virtual DataPort allows you to define relationships between the elements of two views. The following example shows an association where the elements of the `country` view can be related with the elements of the `city` view. Every country is related with zero or more cities.

The association can be seen in the Service Metadata Document:

```
<Association Name="country_city">
  <Documentation>
    <Summary></Summary>
  </Documentation>
  <End Type="com.denodo.odata2.country" Multiplicity="1" Role="country"/>
  <End Type="com.denodo.odata2.city" Multiplicity="*" Role="cities"/>
  <ReferentialConstraint>
    <Principal Role="country">
      <PropertyRef Name="country.country_id"/>
    </Principal>
    <Dependent Role="cities">
      <PropertyRef Name="city.country_id"/>
    </Dependent>
  </ReferentialConstraint>
</Association>
```

6.2   QUERYING ASSOCIATED ENTRIES

When there are associations between entities defined in VDP, it is possible to navigate those associations in order to get all the entries associated with a particular entry. This is done by means of **navigation properties**, a special kind of property created in OData for this specific purpose (allowing the navigation of declared associations).

```
/denodo-odata.svc/<DBNAME>/collectionName(key)/navigationPropertyName
```

Example, being `cities` a navigation property in the `country` entity type (VDP view) that navigates an association towards the `city` entity type (another VDP view).

```
/denodo-odata.svc/movies/country(1)/cities
```

Response:
```
{
 "d": {
   "results": [
     {
       "__metadata": {
         "id": "http://localhost:8080/denodo-odata.svc/movies/city(251)",
         "uri": "http://localhost:8080/denodo-odata.svc/movies/city(251)",
         "type": "com.denodo.odata2.city"
       },
       "city_id": 251,
       "city": "Kabul",
       "country_id": 1,
       "last_update": "\\/Date(1139975125000+0060)\\/",
       "country": {
         "__deferred": {
                                     "uri":    "http://localhost:8080/denodo-
odata.svc/movies/city(251)/country"
         }
       } } ] } }
```

6.3   LISTS OF LINKS BETWEEN ENTRIES

You can get the links to the collection of entries relationed with one in particular:

```
/denodo-odata.svc/<DBNAME>/collectionName(key)/$links/navProp
```

The URL specified above will show all the URLs which allows the access to every entry associated with the entry collectionName(key) through the navigation property navProp.

Example:

```
/denodo-odata.svc/movies/country(2)/$links/cities
```

Response:

```
{
  "d": [
    {
      "uri": "http://localhost:8080/denodo-odata.svc/movies/city(59)"
    },
    {
```

```
      "uri": "http://localhost:8080/denodo-odata.svc/movies/city(63)"
    },
    {
      "uri": "http://localhost:8080/denodo-odata.svc/movies/city(483)"
    }
  ]
}
```

7 SELECTION, PROJECTION, ORDERING AND INCLUDING RELATED RESOURCES

The data returned by the OData service can be controlled to some extent. To achieve this management there are several system query options that are prefixed with a $ character.

7.1 SELECTION: $FILTER

A URI with a `$filter` system query option identifies a subset of the entries from the collection of entries identified by the resource path section of the URI. This subset must satisfy the predicate expression specified by the query option.

The filter query option supports references to properties and literals. The latter can be strings (enclosed in single quotes), numbers, boolean or dates with time zone values (using the literal `datetimeoffset` and the date enclosed in single quotes, e.g. `datetimeoffset'2016-06-06T00:00:00-02:00'`).

In addition to the logical operators (see the **Logical operators section**), the string functions (see the **String functions section**), the math functions (see the **Math functions section**) and the date functions (see the **Date functions section**) explained below, you can use parenthesis to denote precedence by grouping.

7.1.1.1 Logical operators

Operator	Description	Example
Eq	Equal	`/actor?$filter=first_name eq 'GRACE'`
Ne	Not equal	`/actor?$filter=first_name ne 'GRACE'`
Gt	Greater than	`/actor?$filter=actor_id gt 5`
Ge	Greater than or equal	`/actor?$filter=actor_id ge 5`
Lt	Less than	`/actor?$filter=actor_id lt 10`
Le	Less than or equal	`/actor?$filter=actor_id le 10`
And	Logical and	`/actor?$filter=actor_id gt 5 and actor_id lt 10`
Or	Logical or	`/actor?$filter=actor_id lt 5 or first_name eq 'GRACE'`
Not	Logical negation	`/actor?$filter=not (actor_id eq 1)`

7.1.1.2 String functions
There are eleven string functions available that allow managing strings:

- `substringof(string p0, string p1)` returns true when the value of the property name specified in p1 contains the string p0. Otherwise returns false.
- `startswith(string p0, string p1)` returns true when the value of the property name specified in p0 starts with the string p1. Otherwise returns false.

- endswith(string p0, string p1) returns true when the value of the property name specified in p0 ends with the string p1. Otherwise returns false.
- indexof(string p0, string p1) returns the position of the string p1 in the value of the property name specified in p0.
- length(p0) returns the length of the value of the property name specified in p0.
- substring(string p0, int pos) returns a new string that is a substring of the value of the property name specified in p0. The substring begins with the character at the specified pos and extends to the end of this string.
- substring(string p0, int pos, int length) returns a new string that is a substring of the value of the property name specified in p0. The substring begins at the specified pos and extends to the character at index pos + length.
- tolower(string p0) returns a copy of the value of the property name specified in p0 converted to lowercase.
- toupper(string p0) returns a copy of the value of the property name specified in p0 converted to uppercase.
- trim(string p0) returns a copy of the value of the property name specified in p0 with leading and trailing whitespace omitted.
- concat(string p0, string p1) returns a new string that is a concatenation of the string p0 and the string p1.

The following table shows a summary and examples of this functions:

Function	Example
bool substringof(string p0, string p1)	/actor?$filter=substringof('LO', first_name) eq true
bool startswith(string p0, string p1)	/actor?$filter=startswith(first_name, 'JO') eq true
bool endswith(string p0, string p1)	/actor?$filter=endswith(first_name,'ER') eq true
int indexof(string p0, string p1)	/actor?$filter=indexof(last_name, 'LO') eq 3
int length(string p0)	/actor?$filter=length(first_name) eq 4
string substring(string p0, int pos)	/actor?$filter=substring(first_name, 2) eq 'RO'
string substring(string p0, int pos, int length)	/actor?$filter=substring(first_name, 2,3) eq 'TTH'
string tolower(string p0)	/actor?$filter=tolower(first_name) eq 'nick'
string toupper(string p0)	/actor?$filter=toupper(first_name) eq 'NICK'
string trim(string p0)	/actor?$filter=trim(first_name) eq 'JENNIFER'
string concat(string p0, string p1)	/actor?$filter=concat(concat(first_name,', '), last_name) eq 'JENNIFER, DAVIS'

7.1.1.3 Math functions

There are three math functions: `round`, `floor`, `ceiling`. Each one allows double or decimal types as parameters and the returned value is the same as the parameter.

Function	Example
double round(double p0)	`/film?$filter=round(replacement_cost) eq 21`
decimal round(decimal p0)	`/film?$filter=round(rental_rate) eq 1`
double floor(double p0)	`/film?$filter=floor(replacement_cost) eq 20`
decimal floor(decimal p0)	`/film?$filter=floor(rental_rate) eq 0`
double ceiling(double p0)	`/film?$filter=ceiling(replacement_cost) eq 21`
decimal ceiling(decimal p0)	`/film?$filter=ceiling(rental_rate) eq 1`

7.1.1.4 Date functions

Function	Example
int hour(DateTime p0)	`/actor?$filter=hour(last_update) eq 3`
int minute(DateTime p0)	`/actor?$filter=minute(last_update) eq 34`
int second(DateTime p0)	`/actor?$filter=second(last_update) eq 33`

7.2 PROJECTION: $SELECT

A data service URI with a `$select` system query option specifies that a response from the service should return a subset of the properties that are written as a comma-separated list of selection clauses. Each selection clause may be a property name or the * character, which will show all the properties for the resource determined by the resource path section of the URI. Also note that complex properties are forbidden as selection clauses.

Example:

`/denodo-odata.svc/movies/actor?$select=actor_id,first_name,last_name`

Response:

```
...
{
  "__metadata": {
  "id": "http://localhost:8080/denodo-odata.svc/movies/actor(1)",
  "uri": "http://localhost:8080/denodo-odata.svc/movies/actor(1)",
  "type": "com.denodo.odata2.actor"
  },
  "actor_id": 1,
  "first_name": "PENELOPE",
  "last_name": "GUINESS"
},
...
```

Another example:

`/denodo-odata.svc/movies/actor?$select=*`

Response:

```
...
{
  "__metadata": {
    "id": "http://localhost:8080/denodo-odata.svc/movies/actor(1)",
    "uri": "http://localhost:8080/denodo-odata.svc/movies/actor(1)",
    "type": "com.denodo.odata2.actor"
  },
  "actor_id": 1,
  "first_name": "PENELOPE",
  "last_name": "GUINESS",
  "last_update": "\\/Date(1139974473000+0060)\\/"
},
...
```

7.3   ORDERING RESULTS: $ORDERBY

If the URL contains the `$orderby` query string option it specifies the attributes that are used to order the collection of entries identified by the resource path section of the URI:

`/denodo-odata.svc/<DBNAME>/collectionName?$orderby=attribute     [asc| desc]`

To order the collection the resource path must identified a collection of entries, otherwise this option is unavailable.

The optional keywords `asc` and `desc` determine the direction of the sort (ascending or descending, respectively). Default value is `asc`, therefore sorting will be ascending if you do not use one of these keywords.

You can also sort by multiple attributes:

`/denodo-odata.svc/<DBNAME>/collectionName?$orderby=attribute1     [asc| desc],attribute2 [asc|desc]`

Example:

`/denodo-odata.svc/movies/address?$orderby=zip,client_identifier desc`

7.4   INCLUDING RELATED RESOURCES: $EXPAND

This feature is available since Denodo Platform 6.0 and it allows you to specify a $expand system query option in the URI to define the related resources to be included in line with retrieved resources.

The syntax of this query option is a comma-separated list of navigation properties. Additionally each Navigation Property can be followed by a forward slash and another navigation property to enable identifying a multi-level relationship.

Example:

```
/denodo-odata.svc/movies/country?$expand=cities
```

Response:

```
...
"__metadata": {
  "id": "http://localhost:8080/denodo-odata.svc/admin/country(1)",
  "uri": "http://localhost:8080/denodo-odata.svc/admin/country(1)",
  "type": "com.denodo.odata2.country"
},
"country_id": 1,
"country": "Afghanistan",
"last_update": "/Date(1140007440000+0060)/",
"cities": {
 "results": [
    {
      "__metadata": {
        "id": "http://localhost:8080/denodo-odata.svc/admin/city(251)",
        "uri": "http://localhost:8080/denodo-odata.svc/admin/city(251)",
        "type": "com.denodo.odata2.city"
      },
      "city_id": 251,
      "city": "Kabul",
      "country_id": 1,
      "last_update": "/Date(1140007525000+0060)/",
      "country": {
        "__deferred": {
          "uri": "http://localhost:8080/denodo-odata.svc/admin/city(251)/country"
        }
      }
    }
  ]
}
...
```

Example with two navigation properties. This URI identifies the film set as well as the film_actor and the film_categories associated with each film:

```
/denodo-odata.svc/movies/film?$expand=film_actor,film_categories
```

Response:

```
...
"__metadata": {
  "id": "http://localhost:8080/denodo-odata.svc/admin/film(1)",
  "uri": "http://localhost:8080/denodo-odata.svc/admin/film(1)",
  "type": "com.denodo.odata2.film"
},
"film_id": 1,
"title": "ACADEMY DINOSAUR",
"description": "A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher
in The Canadian Rockies",
"release_year": "/Date(1136106000000+0060)/",
"language_id": 1,
"original_language_id": null,
"rental_duration": 6,
"rental_rate": "0.99",
"length": 86,
"replacement_cost": "20.99",
"rating": "PG",
"special_features": "Deleted Scenes,Behind the Scenes",
"last_update": "/Date(1140008622000+0060)/",
"film_actor": {
  "results": [
    {
      "__metadata": {
                                            "id":           "http://localhost:8080/denodo-
odata.svc/admin/film_actor(actor_id=1,film_id=1)",
```

```
                                                       "uri":        "http://localhost:8080/denodo-
odata.svc/admin/film_actor(actor_id=1,film_id=1)",
      "type": "com.denodo.odata2.film_actor"
    },
    "actor_id": 1,
    "film_id": 1,
    "last_update": "/Date(1140008703000+0060)/",
    "actor_role": {
     "__deferred": {
                                                   "uri":        "http://localhost:8080/denodo-
odata.svc/admin/film_actor(actor_id=1,film_id=1)/actor_role"
     }

...

"film_categories": {
  "results": [
    {
     "__metadata": {
                                                   "id":        "http://localhost:8080/denodo-
odata.svc/admin/film_category(film_id=1,category_id=6)",
                                                   "uri":        "http://localhost:8080/denodo-
odata.svc/admin/film_category(film_id=1,category_id=6)",
      "type": "com.denodo.odata2.film_category"
    },
    "film_id": 1,
    "category_id": 6,
    "last_update": "/Date(1140008829000+0060)/",
    "film": {
     "__deferred": {
                                                   "uri":        "http://localhost:8080/denodo-
odata.svc/admin/film_category(film_id=1,category_id=6)/film"
     }
    }
   }
  ]
}

...
```

Example with a multi-level relationship. This URI identifies the film set as well as each of the film_actor associated with each film. In addition, it also identifies the actors associated with each film_actor:

/denodo-odata.svc/movies/film?$expand=film_actor/actor

Response:

```
"__metadata": {
  "id": "http://localhost:8080/denodo-odata.svc/admin/film(1)",
  "uri": "http://localhost:8080/denodo-odata.svc/admin/film(1)",
  "type": "com.denodo.odata2.film"
},
"film_id": 1,
"title": "ACADEMY DINOSAUR",
"description": "A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher
in The Canadian Rockies",
"release_year": "/Date(1136073600000+0060)/",
"language_id": 1,
"original_language_id": null,
"rental_duration": 6,
"rental_rate": "0.99",
"length": 86,
"replacement_cost": "20.99",
"rating": "PG",
"special_features": "Deleted Scenes,Behind the Scenes",
"last_update": "/Date(1140008622000+0060)/",
"film_actor": {
 "results": [
  {
   "__metadata": {
```

```
                                                "id":          "http://localhost:8080/denodo-
odata.svc/admin/film_actor(actor_id=1,film_id=1)",
                                                "uri":          "http://localhost:8080/denodo-
odata.svc/admin/film_actor(actor_id=1,film_id=1)",
    "type": "com.denodo.odata2.film_actor"
  },
  "actor_id": 1,
  "film_id": 1,
  "last_update": "/Date(1140008703000+0060)/",
  "actor": {
   "__metadata": {
     "id": "http://localhost:8080/denodo-odata.svc/admin/actor(1)",
     "uri": "http://localhost:8080/denodo-odata.svc/admin/actor(1)",
     "type": "com.denodo.odata2.actor"
   },
   "actor_id": 1,
   "first_name": "PENELOPE",
   "last_name": "GUINESS",
   "last_update": "/Date(1140006873000+0060)/",
   "film_actor": {
    "__deferred": {
     "uri": "http://localhost:8080/denodo-odata.svc/admin/actor(1)/film_actor"
    }
   }
  },
  "film": {
   "__deferred": {
                                                "uri":         "http://localhost:8080/denodo-
odata.svc/admin/film_actor(actor_id=1,film_id=1)/film"
   }
  }
 },
...
```

7.5 SPECIFYING MAXIMUM NUMBER OF RESULTS: $TOP

With the $top option you can select the n first entries of the collection determined by the resource path section of the URI:

/denodo-odata.svc/<DBNAME>/collectionName?$top=n

Note that n is a positive integer, a negative value means that the URL is malformed.

Example:

/denodo-odata.svc/movies/address?$top=1

Response:

```
{
  "d": {
    "results": [
      {
        "__metadata": {
                                         "id":       "http://localhost:8080/denodo-
odata.svc/movies/address('C001')",
                                         "uri":       "http://localhost:8080/denodo-
odata.svc/movies/address('C001')",
          "type": "com.denodo.odata2.address"
        },
        "client_identifier": "C001",
        "street": "3989 Middlefield Rd",
        "city": "San Jose",
        "zip": "94085",
        "state": "CA",
        "primary_phone": "(408) 813-9318",
```

```
        "country": "UNITED STATES"
      }
    ]
  }
}
```

7.6 SPECIFYING OFFSET: $SKIP

With the option $skip, the n first entries of the collection identified by the resource path section of the URI not will be shown in the response:

denodo-odata.svc/<DBNAME>/collectionName?$skip=n

Note that n is a positive integer, a negative value means that the URL is malformed.

Example:

/denodo-odata.svc/movies/address?$skip=79

Response:

```
{
  "d": {
    "results": [
      {
        "__metadata": {
                                      "id":    "http://localhost:8080/denodo-
odata.svc/movies/address('C080')",
                                      "uri":    "http://localhost:8080/denodo-
odata.svc/movies/address('C080')",
          "type": "com.denodo.odata2.address"
        },
        "client_identifier": "C080",
        "street": "2347 Santa Ana St",
        "city": "Palo Alto",
        "zip": "94303-3141",
        "state": "CA",
        "primary_phone": "(650) 856-9738",
        "country": "UNITED STATES"
      }
    ]
  }
}
```

7.7 ASKING FOR TOTAL RESULT COUNT: $INLINECOUNT

The number of entries in a collection identified by the resource path section of the URI may be added to the data showed in the response using the $inlinecount system query option. This option allow two possible values:

- allpages the OData server include a count of number of entities in the collection identified by the URI.
- none the OData server do not include a count in the response. This option is equivalent to a URI without the $inlinecount option.

Note that the count is calculated after applying any $filter system query options present in the URI.

Examples:

```
/denodo-odata.svc/movies/actor?$inlinecount=allpages
```

Response:

```
{
  "d": {
    "__count": "200",
    "results": [
      …
    ]
  }
}
```

Another example:

```
/denodo-odata.svc/movies/actor?$inlinecount=allpages&$filter=actor_id
eq 1
```

Response:

```
{
  "d": {
    "__count": "1",
    "results": [
      {
        "__metadata": {
          "id": "http://localhost:8080/denodo-odata.svc/movies/actor(1)",
          "uri": "http://localhost:8080/denodo-odata.svc/movies/actor(1)",
          "type": "com.denodo.odata2.actor"
        },
        "actor_id": 1,
        "first_name": "PENELOPE",
        "last_name": "GUINESS",
        "last_update": "\\/Date(1139974473000+0060)\\/"
      }
    ]
  }
}
```

Another example:

```
/denodo-odata.svc/movies/actor?$inlinecount=none
```

Response:

(Actor data, just like without $inlinecount option.)

8 PAGINATION

Whenever the Denodo OData Service has to return a collection of entries which size exceeds that configured at the `server.pageSize` property of its configuration file, it will split the response into pages, returning only the first 1000 entries (the default value for page size).

Denodo OData Service will add to the response a *next link*, which will easily allow the client to request the next page of results.

Next links include a `$skiptoken` parameter. They look like this:

```
/denodo-odata.svc/movies/actor?$skiptoken=1
```

If the original request includes query options, the result will show the *next link* with these query options and a `$skiptoken` as in the examples below.

Example request:

```
/denodo-odata.svc/movies/actor?$top=2
```

Response:
```
...
"__next":                "http://localhost:8080/denodo-odata.svc/movies/actor?
$top=2&$skiptoken=1"
...
```

9 LIMITATIONS

Read-only access

The access to Denodo databases via this Denodo OData Service is read-only.

Representing arrays

OData 2.0 does not support Collections/Bags/Lists that will allow us to give support to arrays as complex objects. Now they are displayed between square brackets as a comma-separated list of values.

Unavailable operators in the $filter system query option

Denodo OData Service does not support arithmetic operators: `add` (addition), `sub` (subtraction), `mul` (multiplication), `div` (division) and `mod` (modulo).

Unavailable functions in the $filter system query option

Denodo OData Service does not provide support for the string function `replace`:
- `string replace(string p0, string find, string replace)`

Denodo OData Service does not provide support for date functions:
- `int day(DateTime p0)`
- `int month(DateTime p0)`
- `int year(DateTime p0)`

Denodo OData Service does not provide support for type functions:
- `bool IsOf(type p0)`
- `bool IsOf(expression p0, type p1)`

Navigation properties in the select system query option

Denodo OData Service does not allow navigation properties as selection clauses.

Navigation using complex properties

When there is an association where one of the elements of an end point is a field of a register, Denodo Virtual DataPort does not allow the navigation from the end point with the complex property.

Order by using complex properties

Denodo Virtual DataPort does not support the order by clause using fields of registers, therefore the `$orderby` query string option is not available if you want to sort by a field of a complex property

9.1 LIMITATIONS IN VERSIONS PRIOR TO DENODO PLATFORM 6.0

Unavailable $expand system query option

This system query option indicates that entries associated with the elements identified by the resource path section must be represented inline.

Counting entries of a collection when there is navigation

Navigational queries does not allow the `count(*)` function, then if the URL contains a navigation the option of counting entries in a collection (`$count`) is not available.