Denodo OData 4.0 Service - User Manual

Revision 20160809

CONTENTS

1 OVERVIEW

OData (**Open Data Protocol**) is a REST-based protocol for querying and updating data using simple HTTP messages. It is an OASIS standard based on technologies such as HTTP, Atom/XML and JSON.

Denodo OData Service allows users to connect to the Denodo Platform and query its databases using an OData 4.0 interface.

Previous version of the Denodo OData Service was interoperable with OData 2.0. From OData 2.0 several features were added, other were improved, pruned, ... Denodo OData Service 4 addresses these changes.

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-odata4-service-<VERSION>.war`
● A documentation folder containing this user manual
● A scripts folder containing the scripts to install the service into the Denodo embedded web container

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 required. And to include the Virtual DataPort (VDP) driver in the web container, in the case of Apache Tomcat 7+ in the folder `/lib`.

VDP driver (`denodo-vdp-jdbcdriver.jar`) is in the folder:

● `$DENODO_HOME\lib\vdp-jdbcdriver-core` in Denodo Platform 5.5
● `$DENODO_HOME\tools\client-drivers\jdbc` in Denodo Platform 6.0

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

The Denodo OData Service has some properties:

● `odataserver.address`: service name. It is an optional property, `/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-odata4-service-<VERSION>/denodo-odata.svc/movies`
  will return this kind of URI within the response:
  `https://gw.denodo.com:9000/ODATA/denodo-odata4-service-<VERSION>/denodo-odata.svc/movies`

- server.pageSize: number of returned entries per request (see **Pagination section** for more information)
- enable.adminUser: false if access to the OData service is denied to the admin user.

The war file includes a configuration file:
WEB-INF/classes/configuration.properties that has default values for the properties explained above:

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

Besides, these properties can be provided by the web container via JNDI (see the **Configuring JNDI resources in Apache Tomcat section** for more information).

Properties configured at the web container as JNDI entries have higher precedence than those established at the configuration.properties file.

Once you deployed the war you can use the Denodo OData Service from a web client, using **HTTP Basic Authentication** with VDP-valid credentials. URLs 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 Denodo OData Service to be installed at the ROOT context of the web server.

Kerberos Authentication is also supported when Kerberos authentication is enabled in the Virtual DataPort **6.0** server (see the Configuring browsers for Kerberos section for more information).

2.1 DEPLOYING INTO THE DENODO EMBEDDED WEB CONTAINER

For deploying the service in the internal web container of the Denodo Platform you should follow these steps:

1. Copy the denodo-odata4-service-<VERSION>.war into
<DENODO_HOME>/resources/apache-tomcat/webapps/

2. Create a context xml file for the service. You can use the xml file in the Denodo OData Service distribution as a template: `/scripts/denodo-odata4-service-<VERSION>.xml`.

3. Copy the `denodo-odata4-service-<VERSION>.xml` file into `<DENODO_HOME>/resources/apache-tomcat/conf/DenodoPlatform-<VERSION>/localhost`

4. Create launch scripts for the service. You can use as a template the files located at the `scripts` folder in the Denodo OData Service distribution:
    ○ `odata4_service_startup.bat(.sh)`
    ○ `odata4_service_shutdown.bat(.sh)`

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

5. Copy the launch scripts into `<DENODO_HOME>/bin`.

6. Check in `<DENODO_HOME>/resources/apache-tomcat/conf/catalina.properties` file that the `common.loader` property (`shared.loader` in Denodo 5.5) includes a reference to the VDP JDBC Driver. If missing, add it:

    ● `${catalina.base}/../../tools/client-drivers/jdbc/denodo-vdp-jdbcdriver.jar` for Denodo Platform 6.0

    ● `${catalina.base}/../../lib/vdp-jdbcdriver-core/denodo-vdp-jdbcdriver.jar` for Denodo Platform 5.5

7. Configure via JNDI the VDP data source and those properties you want to change their default value (see the **Configuring JNDI resources in Apache Tomcat** for more information).

8. Run the `<DENODO_HOME>/bin/odata4_service_startup.bat(.sh)` script and navigate to:

    `http://localhost:9090/denodo-odata4-service-<VERSION>/denodo-odata.svc/<DBNAME>`

2.2 CONFIGURING JNDI RESOURCES IN APACHE TOMCAT

2.2.1 Data source configuration

1. Declare the data source with the name `jdbc/VDPdatabase` in Apache Tomcat's `server.xml` file.

    If you deploy the Denodo OData Service into the Denodo embedded web container you must declare the JNDI resource in:

    ● `<DENODO_HOME>/resources/apache-tomcat/conf/`**`server.xml.template`** file for Denodo Platform 5.5.

- <DENODO_HOME>/resources/apache-tomcat/conf/**server.xml** file for Denodo Platform 6.0

Below, there is an example:

```
<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 in the 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>
```

2.2.2 Properties configuration

Declare the properties values in the `context.xml` file:

```xml
<Context>
  <!-- OPTIONAL PARAMETER -->
  <Environment type="java.lang.String"
               name="odataserver.address"
               value="/denodo-odata.svc"/>


  <!-- OPTIONAL PARAMETER -->
  <Environment type="java.lang.String"
               name="odataserver.serviceRoot"
               value="https://gw.denodo.com:9000/ODATA/"/>


 <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 ENABLING CROSS-ORIGIN RESOURCE SHARING (CORS)

Cross-origin resource sharing (CORS) is a mechanism whereby a browser and server interact to determine whether to allow a web page to perform HTTP requests to a domain other than the domain from which the web page is originated. I.e. so the browser can determine if it is safe that a page served from the domain http://foo.com can send requests to a server in the domain http://bar.com.

Once deployed, to enable the CORS support on the Denodo OData Service, follow these steps:

1. Edit the `web.xml` file which is at the `WEB-INF` folder. If you have deployed the Denodo OData Service into the Denodo embedded web container the path to the directory is `<DENODO_HOME>/resources/apache-tomcat/webapps/denodo-odata4-service-<VERSION>/WEB-INF/`.

2. Add the External Entity adding the elements `DOCTYPE` and `ENTITY` above the root element `web-app`:

```
<!DOCTYPE doc [
<!ENTITY cors_settings SYSTEM "cors_settings.xml">
]>


<web-app ...>
```

   Besides, add the reference in the filters section:

```
<!-- ****************** -->
<!-- Filters         -->
<!-- ****************** -->


&cors_settings;
```

3. Edit the file `cors_settings.xml` which is at the `WEB-INF` folder. If you have deployed the Denodo OData Service into the Denodo embedded web container the path to the directory is `<DENODO_HOME>/resources/apache-tomcat/webapps/denodo-odata4-service-<VERSION>/WEB-INF/`.

4. Uncomment the contents of the file.

5. If the value of the property is *, the Service will allow CORS requests received from any domain.

   To limit the domains from which CORS requests are allowed, change the value of the property `cors.allowOrigin`. Replace * with the list of allowed URLs. Separate each URL by a comma. For example, `https://foo.com`, `http://foo.com`, `https://foo.bar.com`.

   CORS requests from any other origin will be denied with the HTTP code 403 (Forbidden)

   Important: for each URL, enter its protocol as well. URLs that not contain the protocol are invalid. E.g. **foo.com** is invalid.

6. Restart the Denodo OData Service to apply the changes.

Note that if you have deployed the Denodo OData Service in an external web container you need to configure the container to allow external entities to be loaded or, manually replace in `web.xml` file at `WEB-INF` folder `&cors_settings` reference with the content of the file `cors_settings.xml` which is at `WEB-INF` folder and remove the elements `DOCTYPE` and `ENTITY` above the root element `web-app`.

4 FEATURES

OData 4.0 defines three levels of conformance for OData services. OData Denodo Service conforms to the **Intermediate Conformance Level** providing the following functionality:

- **Read-only** access to Denodo databases:
  - Metadata
  - Entities
  - Items of an entity
  - Properties of an item
  - Property values
  - Relationships between entities
- Format results in Atom and JSON
- Query options
  - `$select`
  - `$filter`
  - `$expand`
  - `$orderby`
  - `$skip`
  - `$top`
  - `$count`
- Parameter aliases
- Pagination
- HTTP Authentication
  - Basic
  - Kerberos

5   SERVING METADATA

There are two types of metadata documents:

● The **Service Document** lists all the entities offered by the data service.
   It is available at the root URI of the 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

```
 {
   "@odata.context": "$metadata",
   "value": [
     {
       "name": "actor",
       "url": "actor"
     },
     {
       "name": "address",
       "url": "address"
     },
     {
       "name": "city",
       "url": "city"
     },
     {
       "name": "country",
       "url": "country"
     },
     {
       "name": "film",
       "url": "film"
     },
     {
       "name": "film_actor",
       "url": "film_actor"
     }
   ]
 }
```

● The **Service Metadata Document**, also called **Entity Data Model (EDM)**, is an XML representation of the data model exposed by the service. It is available at .../$metadata:

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

Example of an entity country that has a relationship with zero or more city elements:

```
<EntityType Name="country">
  <Key>
      <PropertyRef Name="country_id"/>
  </Key>
  <Property Name="country_id" Type="Edm.Int16"
            Nullable="false"/>
  <Property Name="country" Type="Edm.String"
            MaxLength="50"/>
  <Property Name="last_update" Type="Edm.DateTimeOffset"
            Precision="19"/>
  <NavigationProperty Name="cities"
              Type="Collection(com.denodo.odata4.city)"
              Partner="country">
      <ReferentialConstraint Property="country_id"
              ReferencedProperty="country_id"/>
  </NavigationProperty>
</EntityType>
```

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*
Relationship	**Association Definition**
Entity Set	**View Data**
Imported Functions	**-**

6 QUERYING DATA: THE BASICS

6.1 QUERYING COLLECTIONS

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

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

Example:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    },
     …
    {
      "actor_id": 1000,
      "first_name": "CHRISTIAN",
      "last_name": "GABLE",
      "last_update": "2006-02-15T11:34:33Z"
    }
  ],
  "@odata.nextLink": "/denodo-odata.svc/movies/actor?$skiptoken=1"
}
```

Note that, for the sake of simplicity, we are removing the server and port from the examples.

6.2 OBTAINING ITEMS BY PRIMARY KEY

Each item 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 primary key can be a compound key, and in that 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.

6.3   ACCESSING INDIVIDUAL PROPERTIES

Properties of an item can be accessed individually:

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

Example:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata
                     #actor/first_name",
  "value": "PENELOPE"
}
```

6.4   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
```

6.5 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(key)/propName/complexProp/propName
```

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

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata
                                        #struct_table_film/$entity",
  "table_id": 1,
  "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:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata
                                    #struct_table_film/title",
  "value": "ACADEMY DINOSAUR"
}
```

6.6 COUNTING ELEMENTS IN A COLLECTION: $COUNT

If you want to know the number of elements 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
```

6.7 ESTABLISHING THE RESPONSE FORMAT

OData resources can be represented in Atom or JSON, being JSON the default format.

Clients can request the format through the system query option $format or through the Accept header. In the case that both the Accept header and the $format query option are specified on a request, the $format query option has a higher precedence.

With the system query option $format:

● /denodo-odata.svc/movies/actor?$format=json shows data using the JSON format as defined in OData:JSON
   Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    },
    …
    {
      "actor_id": 1000,
      "first_name": "CHRISTIAN",
      "last_name": "GABLE",
      "last_update": "2006-02-15T11:34:33Z"
    }
  ],
  "@odata.nextLink": "/denodo-odata.svc/movies/actor?$skiptoken=1"
```

- /denodo-odata.svc/movies/actor?$format=atom shows data using the Atom format defined in OData:Atom.
  Response:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<a:feed xmlns:a="http://www.w3.org/2005/Atom" xmlns:m="http://docs.oasis-
open.org/odata/ns/metadata" xmlns:d="http://docs.oasis-
open.org/odata/ns/data" m:context="/denodo-odata.svc/movies/
$metadata#actor">
    <a:id>/denodo-odata.svc/movies/actor</a:id>
    <a:link rel="next" href="/denodo-odata.svc/movies/actor?
$format=atom&amp;$skiptoken=1"/>
    <a:entry>
        <a:id>/denodo-odata.svc/movies/actor(1)</a:id>
        <a:title/>
        <a:summary/>
        <a:updated>2016-04-22T13:07:42Z</a:updated>
        <a:author>
            <a:name/>
        </a:author>
        <a:link rel="edit" href="/denodo-odata.svc/movies/actor(1)"/>
        <a:link rel="http://docs.oasis-open.org/odata/ns/related/actors"
type="application/atom+xml;type=feed" title="actors" href="/denodo-
odata.svc/movies/actor(1)/actors"/>
        <a:category scheme="http://docs.oasis-open.org/odata/ns/scheme"
term="#com.denodo.odata4.actor"/>
        <a:content type="application/xml">
            <m:properties>
                <d:actor_id m:type="Int16">1</d:actor_id>
                <d:first_name>PENELOPE</d:first_name>
                <d:last_name>GUINESS</d:last_name>
                <d:last_update m:type="DateTimeOffset">2006-02-
15T11:34:33Z</d:last_update>
            </m:properties>
        </a:content>
    </a:entry>
...
```

With the HTTP header Accept:

```
Accept: application/atom+xml
```

7 NAVIGATING ASSOCIATIONS

7.1 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`.

This relationship is represented in the Service Metadata Document as a **navigation property**:

```
<EntityType Name="country">
    <Key>
        <PropertyRef Name="country_id"/>
    </Key>
    <Property Name="country_id" Type="Edm.Int16"
            Nullable="false"/>
    <Property Name="country" Type="Edm.String" MaxLength="50"/>
    <Property Name="last_update" Type="Edm.DateTimeOffset"
            Precision="19"/>
    <NavigationProperty Name="cities"
                        Type="Collection(com.denodo.odata4.city)"
                        Partner="country">
        <ReferentialConstraint Property="country_id"
                            ReferencedProperty="country_id"/>
    </NavigationProperty>
</EntityType>
```
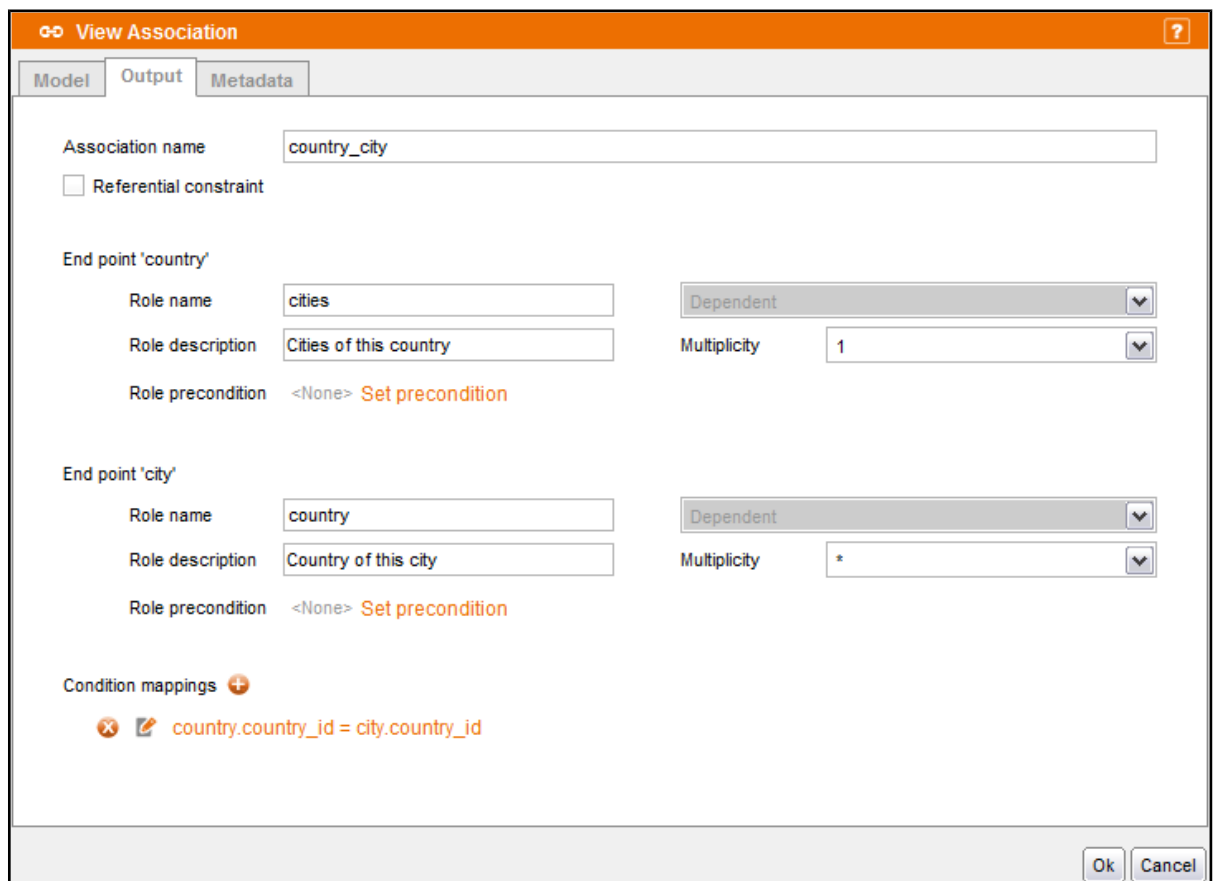
7.2 QUERYING RELATED ENTRIES

Denodo OData Service allows navigating the associations defined in VDP in order to get all the entities related with a particular entity. This is done by means of **navigation properties**.

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

For example, being `cities` a navigation property in the entity type `country` that navigates towards the `city` entity type:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#city",
  "value": [
    {
      "city_id": 59,
      "city": "Batna",
      "country_id": 2,
      "last_update": "2006-02-15T11:45:25Z"
    },
    {
      "city_id": 63,
      "city": "Bchar",
      "country_id": 2,
      "last_update": "2006-02-15T11:45:25Z"
    },
    {
      "city_id": 483,
      "city": "Skikda",
      "country_id": 2,
      "last_update": "2006-02-15T11:45:25Z"
    }
  ]
}
```

7.3 REFERENCING RELATED ENTITIES

You can get the references to the collection of entities (instead of the actual entities) related with one particular entity.

This is requested appending `$ref` to the URL path. The old syntax `/$links/` has been dropped in OData 4.

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

The URL specified below will show the references of every `city` associated with the entry `country(2)` through the navigation property `cities`:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#cities",
  "value": [
    {
      "@odata.id": "city(59)"
    },
    {
      "@odata.id": "city(63)"
    },
    {
      "@odata.id": "city(483)"
    }
  ]
}
```

8 ADVANCED QUERYING

OData defines some query options that allows refining the requests: $filter, $select, $orderby and $expand.

8.1 SELECTION: $FILTER

A URI with a $filter system query option identifies a subset of the entries from the collection that satisfy the $filter predicate expression.

Expressions can reference properties and literals. The latter can be strings (enclosed in single quotes), the null literal, numbers or boolean values.

Denodo OData service supports the following operations and functions:

8.1.1 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?$filter=not (actor_id eq 1)
add	Addition	/film?$filter=length add 30 gt 180
sub	Subtraction	/film?$filter=length sub 30 gt 120
mul	Multiplication	/film?$filter=length mul 2 ge 300
div	Division	/film?$filter=length div 3 eq 60

mod	Modulo	/film?$filter=length mod 10 eq 8
()	Precedence grouping	/actor?$filter=actor_id lt 7 and (first_name eq 'NICK' or actor_id gt 3)

8.1.2 String functions

The following functions are available for strings operations:

- `contains(string p0, string p1)` returns `true` when the value of the property name specified in `p0` contains the string `p1`. 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 these functions:

Function	Example
bool contains(string p0, string p1)	/actor?$filter=contains(first_name, 'LO')
bool startswith(string p0, string p1)	/actor?$filter=startswith(first_name, 'JO')
bool endswith(string p0, string p1)	/actor?$filter=endswith(first_name,'ER')

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'`

8.1.3 Math functions

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

Function	Example
round	`/film?$filter=round(replacement_cost) eq 21`
floor	`/film?$filter=floor(replacement_cost) eq 20`
ceiling	`/film?$filter=ceiling(replacement_cost) eq 21`

8.1.4 Date functions

Function	Example
int year(DateTimeOffset p0)	`/actor?$filter=year(last_update) eq 2016`
int month(DateTimeOffset p0)	`/actor?$filter=month(last_update) eq 12`
int day(DateTimeOffset p0)	`/actor?$filter=day(last_update) eq 31`
int hour(DateTimeOffset p0)	`/actor?$filter=hour(last_update) eq 3`
int minute(DateTimeOffset p0)	`/actor?$filter=minute(last_update) eq 34`
int second(DateTimeOffset p0)	`/actor?$filter=second(last_update) eq 33`
DateTimeOffset now()	`/actor?$filter=last_update lt now()`

8.2   PROJECTION: $SELECT

The $select system query option returns only the properties explicitly requested.
$select expression can be a comma-separated lists of properties or the star operator
(*), which will retrieve all the properties.
Example:

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

Response:

```
{
  "@odata.context":"/denodo-odata.svc/movies/$metadata#actor
                        (actor_id,first_name,last_name)",
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS"
    },
...
```

Another example:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor(*)",
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    },
...
```

Note that complex properties can be used in $select expressions:

```
denodo-odata.svc/admin/struct_table_film?$select=film_data/title
```

Response:

```
  "@odata.context":"/denodo-odata.svc/admin/$metadata
                            #struct_table_film(film_data/title)",
  "value": [
    {
      "@odata.id": "/denodo-odata.svc/admin/struct_table_film(1)",
      "film_data": {
        "title": "ACADEMY DINOSAUR"
      }
    },
...
```

8.3 ORDERING RESULTS: $ORDERBY

The $orderby query string option specifies the order in which items are returned:

```
/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 keywords asc and desc determine the direction of the sort (ascending or descending, respectively). If asc or desc are not specified items are returned in ascending order. Null values come before non-null values when sorting in ascending order and vice versa.

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
```

8.4 INCLUDING RELATED RESOURCES: $EXPAND

$expand is **available since Denodo Platform 6.0**.

An $expand expression is a comma-separated list of navigation properties that specifies the related entities that should be represented inline.

Example:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#country",
  "value": [
    {
      "country_id": 1,
      "country": "Afghanistan",
      "last_update": "2006-02-15T11:44:00Z",
      "city": [
        {
          "city_id": 251,
          "city": "Kabul",
          "country_id": 1,
          "last_update": "2006-02-15T11:45:25Z"
        }
      ]
    },
...
```

The following is an example with two navigation properties. This URI identifies the film set as well as the `film_actor` (actors is the navigation property) and the `film_category` (categories is the navigation property) associated with each film:

```
/denodo-odata.svc/movies/film?$expand=actors,categories
```

Response:

```
...
  {
    "film_id": 2,
    "title": "ACE GOLDFINGER",
    "description": "A Astounding Epistle of a Database
                   Administrator And a Explorer who must Find a
                   Car in Ancient China",
    "release_year": "2005-12-31T23:00:00Z",
    "language_id": 1,
    "original_language_id": null,
    "rental_duration": 3,
    "rental_rate": 4.99,
    "length": 48,
    "replacement_cost": 12.99,
    "rating": "G",
    "special_features": "Trailers,Deleted Scenes",
    "last_update": "2006-02-15T12:03:42Z",
```

```
      "categories": [
        {
          "film_id": 2,
          "category_id": 11,
          "last_update": "2006-02-15T12:07:09Z"
        }
      ],
      "actors": [
        {
          "actor_id": 19,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z"
        },
        {
          "actor_id": 85,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z"
        },
        {
          "actor_id": 90,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z"
        },
        {
          "actor_id": 160,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z"
        }
      ]
    },
...
```

Expanded entities can be filtered, ordered, paged, projected and expanded. Allowed system query options are $filter, $select, $orderby, $skip, $top, $count and $expand. These expand options are expressed as a semicolon-separated list enclosed in parentheses:

```
/denodo-odata.svc/movies/film?$expand=categories($select=last_update)
```

```
/denodo-odata.svc/movies/film?$expand=categories($orderby=last_update
asc)
```

```
/denodo-odata.svc/movies/film?$expand=actors($count=true)
```

```
/denodo-odata.svc/movies/film?$expand=actors($expand=categories)
```

Response:

```
...
  {
      "film_id": 2,
      "title": "ACE GOLDFINGER",
      "description": "A Astounding Epistle of a Database Administrator
                     And a Explorer who must Find a Car in Ancient
                     China",
      "release_year": "2005-12-31T23:00:00Z",
      "language_id": 1,
      "original_language_id": null,
      "rental_duration": 3,
      "rental_rate": 4.99,
      "length": 48,
      "replacement_cost": 12.99,
      "rating": "G",
      "special_features": "Trailers,Deleted Scenes",
      "last_update": "2006-02-15T12:03:42Z",
      "actors": [
        {
          "actor_id": 19,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z",
          "categories": [
            {
              "film_id": 2,
              "category_id": 11,
              "last_update": "2006-02-15T12:07:09Z"
            }
          ]
        },
        {
          "actor_id": 85,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z",
          "categories": [
            {
              "film_id": 2,
              "category_id": 11,
              "last_update": "2006-02-15T12:07:09Z"
            }
          ]
        },
        {
          "actor_id": 90,
          "film_id": 2,
          "last_update": "2006-02-15T12:05:03Z",
          "categories": [
            {
              "film_id": 2,
              "category_id": 11,
              "last_update": "2006-02-15T12:07:09Z"
            }
          ]
        },
        {
          "actor_id": 160,
```

```
            "film_id": 2,
            "last_update": "2006-02-15T12:05:03Z",
            "categories": [
              {
                "film_id": 2,
                "category_id": 11,
                "last_update": "2006-02-15T12:07:09Z"
              }
            ]
          }
        ]
      },
...
```

8.5   PARAMETER ALIASES

Parameter aliases are identifiers prefixed with an @ sign. They can be used in query expressions to avoid stating the same literal multiple times, or deferring lengthy literals to a place where they are easier to read.

Example:
```
/denodo-odata.svc/movies/film?$filter=contains(title,@p1) and
    not contains(description,@p1)&@p1='ACADEMY DINOSAUR'
```

8.6   SPECIFYING MAXIMUM NUMBER OF RESULTS: $TOP

With the $top option you can select the n first entries of the collection, being n a non-negative integer:

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

Example:
```
/denodo-odata.svc/movies/actor?$top=1
```

Response:
```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    }
  ]
}
```

8.7 SPECIFYING OFFSET: $SKIP

With the option $skip, the n first entries of the collection will not be shown in the response. n is a non-negative integer.

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

Example:

`/denodo-odata.svc/movies/actor?$skip=199`

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "value": [
    {
      "actor_id": 200,
      "first_name": "THORA",
      "last_name": "TEMPLE",
      "last_update": "2006-02-15T11:34:33Z"
    }
  ]
}
```

8.8 ASKING FOR TOTAL RESULT COUNT: $COUNT

The $count system query option returns the number of items returned in the response along with the result.

The old syntax $inlinecount=allpages has been shortened in OData 4 to $count=true.

The $count system query option ignores $top, $skip, and $expand query options, and returns the total count of results across all pages including only those results matching any specified $filter .

Examples:

`/denodo-odata.svc/movies/actor?$count=true`

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "@odata.count": 200,
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    },
...
```

Another example:

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

Response:

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "@odata.count": 1,
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    }
  ]
}
```

Another example:

```
/denodo-odata.svc/movies/actor?$count=false
```

Response:

Actor data, just the same as a request without $count option.

```
{
  "@odata.context": "/denodo-odata.svc/movies/$metadata#actor",
  "value": [
    {
      "actor_id": 1,
      "first_name": "PENELOPE",
      "last_name": "GUINESS",
      "last_update": "2006-02-15T11:34:33Z"
    },
...
```

9 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?$top=2&$skip=1
```

Response:

```
...
"@odata.nextLink":"/denodo-odata.svc/movies/actor?
$top=2&$skip=1&$skiptoken=1"
...
```

Besides using the configuration property `server.pageSize` users can request another page size using the `odata.maxpagesize` preference in the `Prefer` header.

10 CONFIGURING BROWSERS FOR KERBEROS

Because the browser is connecting to the Denodo OData Service which is communicating with a VDP server, Kerberos authentication and delegation should be enabled for the browser.

10.1 FIREFOX

By default Firefox does not enable SPNEGO authentication, and consequently Kerberos.

10.1.1 Enable Kerberos Authentication

1. Go to **about:config** in the address bar in Firefox
2. Click **I'll be careful, I promise** when warned about changing advanced settings
3. Enter **negotiate** in the **Search** box
4. Set value of the **network.negotiate-auth.trusted-uris** to your domain name

10.1.2 Enable Kerberos Delegation

1. Go to **about:config** in the address bar in Firefox
2. Click **I'll be careful, I promise** when warned about changing advanced settings
3. Enter **negotiate** in the **Search** box
4. Set value of the **network.negotiate-auth.delegation-uris** to your domain name

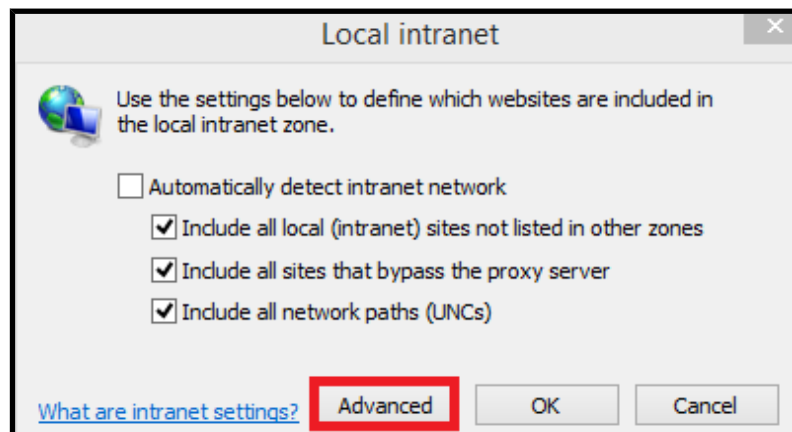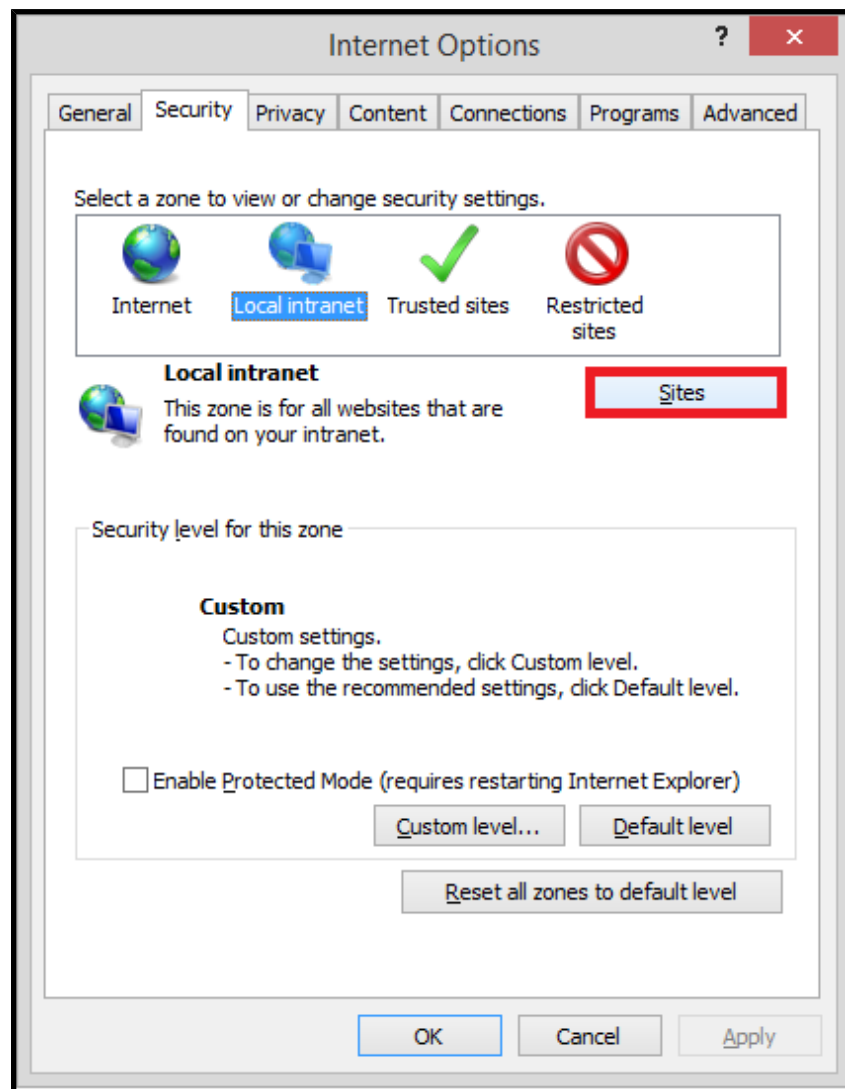Restart browser and see if everything works.

10.2 INTERNET EXPLORER

10.2.1 Enable Kerberos Authentication

1. Click **Tools -> Internet Options**
2. **Advanced** tab
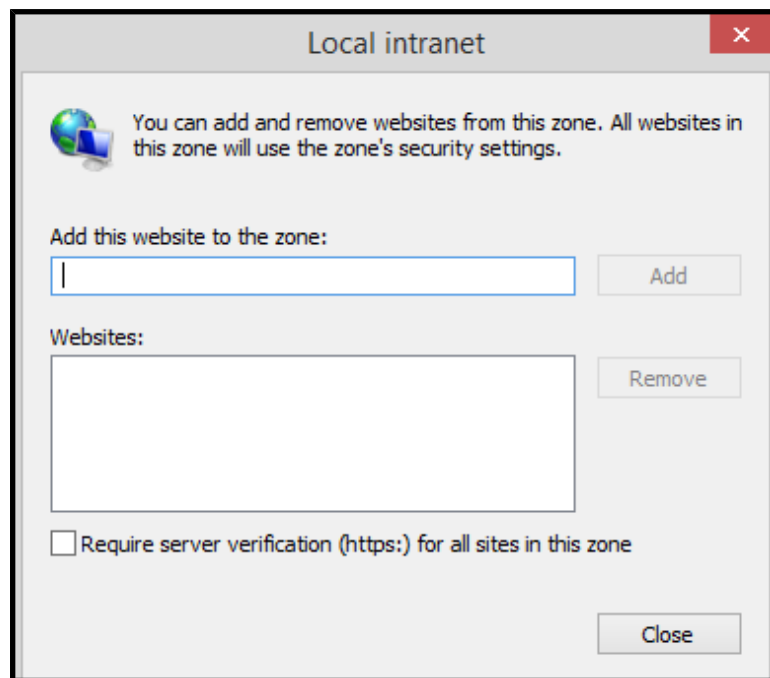3. Enable checkbox for **Enable Integrated Windows Authentication**
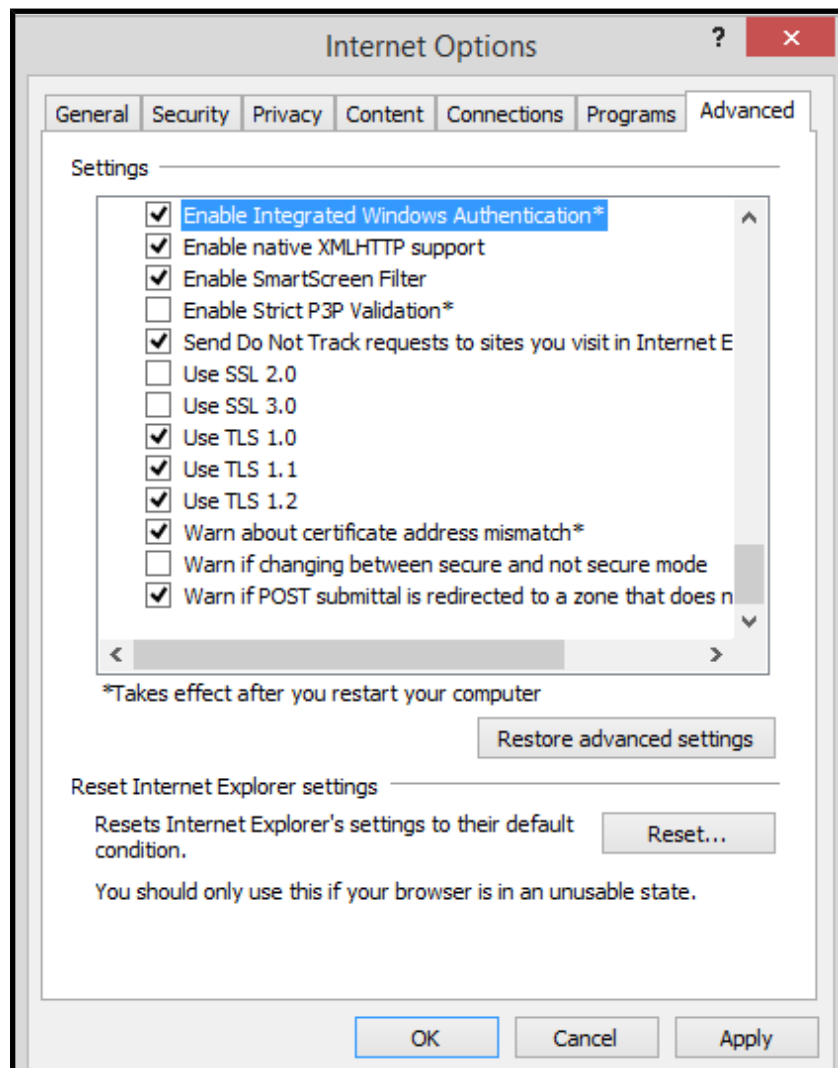
10.2.2 Enable Kerberos Delegation

1. Click **Tools -> Internet Options**
2. **Security** tab -> **Local intranet** -> **Sites**

3. Add the site in question.

Restart browser and see if everything works.

10.3 CHROME

Chrome in Windows will use the Internet Explorer settings, so configure them within Internet Explorer's **Tools -> Internet Options** dialog as explained in the previous section.

11 LIMITATIONS

Read-only access

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

Unavailable functions in the `$filter` system query option

Denodo OData Service does not provide support for the operator `has`.

Denodo OData Service does not provide support for functions:
- `fractionalseconds, date, time, totaloffsetminutes, mindatetime` and `maxdatetime`
- `isof and cast`
- `geo.distance, geo.length and geo.intersects`

Any requests containing these functions will return the error code: `501 Not Implemented`.

Unavailable operators in the `$filter` system query option

Denodo OData Service does not provide support for lambda operators: `any` and `all`.

Unavailable literals in the `$filter` system query option

Denodo OData Service does not provide support for the literal `$it` in expressions to refer to the current instance of the collection identified by the resource path.

Denodo OData Service does not provide support for the literal `$root` in expressions to refer to resources of the same service.

Navigation properties in the `$select` system query option

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

Unavailable `$search` system query option

The `$search` system query option is not available in Denodo OData Service.

Unavailable expand option `$levels`

Denodo OData Service does not provide support for the `$levels` expand option that allows requesting recursive expands.

Unavailable Cross-Join queries

Denodo OData Service does not provide support for the resource `~/crossjoin` to represent Cartesian products of entities.

Resolving references via the resource `$entity`

Denodo OData Service does not provide support for the resource `~/$entity` that allows resolving entity references using the query option `$id`.

White spaces in the URL

Denodo OData Service does not allow white spaces between function parameters and before or after the equal sign (=) that is used to specify query options.

Redundant keys for dependent entities

Denodo OData Service does not allow the omission of redundant keys when there are key properties determined by the parent in a relationship. You have to add all key properties in each level.

Order by with `$expand` system query option

Denodo OData Service does not provide support for the `$orderby` query option inside $expand system query option.

Order by using complex properties

Denodo Virtual DataPort does not support the order by clause using fields of registers, therefore the `$orderby` query option is not available for complex properties.

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.

Referencing expanded entities

Denodo OData Service does not provide support for referencing a single entity in a "to-one" relationship when the related data is requested using the $expand query option and the selected format for the result is JSON.

Below, there is an example of this limitation.

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

Note that this request will fail because JSON is the default format.

11.1 LIMITATIONS IN VERSIONS PRIOR TO DENODO PLATFORM 6.0

Unavailable $expand system query option

An $expand expression is a comma-separated list of navigation properties that specifies the related entities that should be represented inline.

Counting entries of a collection when there is navigation

Navigational queries does not allow the count(*) function. When the URL contains a navigation the option of counting items in a collection ($count) is not available.