

# White Paper

## Fabasoft Integration for OData

2025 July Release

Copyright © Fabasoft R&D GmbH, Linz, Austria, 2025.

All rights reserved. All hardware and software names used are registered trade names and/or registered trademarks of the respective manufacturers.

No rights to our software or our professional services, or results of our professional services, or other protected rights can be based on the handing over and presentation of these documents.

## Contents

<b>1 Introduction</b>	<b>5</b>
<b>2 Installation of the Fabasoft OData Service</b>	<b>5</b>
<b>3 Basic Functionality</b>	<b>5</b>
3.1 Creating an OData Service	5
3.2 Query Types	6
3.2.1 Found Objects	6
3.2.2 Referenced Objects	6
3.2.3 User Query	6
3.2.4 Examples Found Objects vs. Referenced Objects vs. Own Query	7
3.2.5 User Query Together With Filter Options	7
3.3 Filter Options	8
3.4 Other OData Parameters	9
3.4.1 User Query Together With Filter Options	9
3.5 Creating a Password for Applications	10
3.6 Licensing	10
<b>4 Mapping of Fabasoft Cloud Objects to OData Entities</b>	<b>11</b>
4.1 Mapping of Names	11
4.1.1 "Use Programming Names" Option	11
4.2 Attribute Value Generation	12
4.2.1 Basic Types	12
4.2.2 Lists	12
4.2.3 Aggregates	12
<b>5 In-Depth Explanation: Execution of OData Requests</b>	<b>12</b>
5.1 OData GET Request to an OData Service	13
5.2 OData GET Request to an Entity of an OData Service	13
5.2.1 Calculating the Cube	13
5.2.2 Getting the Attribute Values	13
<b>6 Access the OData Service via Microsoft Power BI</b>	<b>13</b>
<b>7 Access the OData Service as a Developer</b>	<b>16</b>
<b>8 Performance</b>	<b>17</b>
8.1 Performance of the Initial Call to an OData Service	17
8.2 Performance of Subsequent Calls to OData Service	17

8.3 Performance Optimization "Getting Only X Rows" .....	18
8.4 Precalculation .....	18
<b>9 Restrictions .....</b>	<b>19</b>
9.1 Data Types.....	19
9.2 Categories and Entities.....	19
9.3 User Forms.....	20

## 1 Introduction

With the Fabasoft Integration for OData (<https://www.odata.org>), structured data can be accessed.

This document describes the setup of an OData service as well as the access via Microsoft Power BI. For detailed instructions on how to generate sophisticated reports with charts, lists, relations, etc., please check out the official Microsoft Power BI documentation.

## 2 Installation of the Fabasoft OData Service

For the time being, the Fabasoft OData service functionality must be enabled by Fabasoft Cloud Support ([cloudsupport@fabasoft.com](mailto:cloudsupport@fabasoft.com)).

## 3 Basic Functionality

An OData service can be created in apps that have reporting enabled (e.g. Scrum). The OData service contains the definition of the object classes and properties that are exposed via the OData API.

Depending on where you create the OData service, various source objects can be used. For example, in order to gain access to your contracts, create an OData service inside your Contract Manager app.

Having created an OData service, you receive a URL which can then be used to access the data via a client like Microsoft Power BI. The OData protocol is a standardized protocol which is understood by any client - even a web browser.

Aside from the OData URL, you require the "Access for Applications" password for authentication.

### 3.1 Creating an OData Service

Once reporting has been enabled in the app of your choice, you need to create the OData service in the context where your data is stored so that it can be queried.

To create an OData service, proceed as follows:

1. Navigate into the desired app configuration or app room, then into the "Reports and OData Services" area.
2. Click the "Create OData Service" action.
3. Define the following settings and click "Next".
  - Name  
The name of the OData service.
  - Language  
Multilingual strings are provided in the defined language in the data source.
  - Use Programming Name  
If enabled, object classes and properties will not be translated and the programming names are used instead.

- Cache Duration  
Defines how long the data is cached. Defining longer time intervals will increase performance during subsequent calls.
- 4. Choose the object class of which you want to add properties to the model for the OData service and click "Next".
- 5. Select the desired properties and click "Next".  
**Note:** You need to consider that adding a great number of properties will require additional resources to load. It is recommended to add the minimum required number of classes and properties.
- 6. Click "Next".
- 7. If needed, you can add further properties of other object classes by clicking "Add Entry".
- 8. Click "Next".
- 9. Copy the provided link to the OData service and click "Next".  
**Note:** You will be able to copy this link later as well by clicking on the generated OData service.

## 3.2 Query Types

For each entity you can select between found objects, referenced objects and user query. For "Found Objects" a query is defined where these objects are searched for. Referenced Objects are just objects referenced by another attribute.

There the following restrictions are applied: for the object classes #Object, #User and #Group only "Referenced Objects" is possible. In general, also the permission #AccTypeSearch must be given to be able to select "Found Objects".

### 3.2.1 Found Objects

For "Found Objects" a predefined query will be executed and all the objects are searched for.

### 3.2.2 Referenced Objects

Referenced objects are objects which are referenced by another object in all the other entities. You could for example, search for ("Referenced Objects") your documents and add there an attribute "Created by". These users can then be found in the "User" entity.

### 3.2.3 User Query

The user query enables you to create own queries which narrows down the found objects.

**Example:**

```
SELECT * FROM COMPONENT@100.200:EntityType WHERE .FSCTEAMROOM@1.1001:objteamroom IN (COO.100.200.300.xxxx,COO.100.200.300.xxxx) AND .objexternalkey LIKE 'x1%'
```

This query would narrow down the objects to a Teamroom and looks through the attribute objexternalkey. The referenced objects will work the same, just with the objects which were filtered with the user query.

### 3.2.4 Examples Found Objects vs. Referenced Objects vs. Own Query

These examples should demonstrate the decision between “Found Objects” and “Referenced Objects”. In general, if you are unsure, “Found Objects” can be a good start as you will get data there. Only if you want to restrict the found objects, switch to the option “Referenced Objects”.

#### Example 1: Books used by Machines

Imagine there are 10 000 registered books in my organization. Some books (like manuals, about 100) are for maintaining my machines.

I want to find with OData my machines and the books used for their maintenance.

So, I define an OData Service with restriction to my whole organization with two entities:

1. Machine with option “Found Objects” because I want that all my machines are found via query.
  - 1.1. I add there the attribute “Manual” which refers to the book used for maintenance.
2. Book with option “Referenced Objects” because I only want to get books which are referenced by the “Manual” attribute from my machines.

The outcome is that the “Machine” entity contains all my machines and the “Book” contains all manuals referenced in my machines (about 100).

#### Example 2: Full list of Books and Machines

In comparison to the first example, I want to do statistics on my inventory. For that I need all machines and all books I have in the Fabasoft Cloud.

So, I define both entities with the option “Found Objects”. The result will be all my machines and all my books (about 10 000). Further filters I could apply with my Business Intelligence Tool like Microsoft Power BI.

#### Example 3: Machines which have been built before 2021

Imagine we want to monitor only machines, which have been built before 2021, “Found Objects” and “Referenced Objects” does not help us because we are always loading all machines.

For such a filtering of the input data, user queries have been introduced as described in chapter 3.2.3 “User Query”.

```
SELECT * FROM COMPONENT@100.200:Machine WHERE .FSCTEAMROOM@1.1001:objteamroom IN (COO.100.200.300.xxxx,COO.100.200.300.xxxx) AND .builtatyear < 2021
```

### 3.2.5 User Query Together With Filter Options

If no user query is provided, a query will be created automatically. The filter options like

```
/Story?$filter=objchangedat gteq 2024-06-28T11:55:46
```

are then added automatically to the query.

But what if you provide a user query? It cannot be determined automatically, where in your user query to add the filter criteria.

#### Example:

You have a user query

```
SELECT * FROM FSCSCRUM@1.1001:Story WHERE .objname LIKE "%Storage%" AND .objname LIKE "%Service%"
```

If you want to request this OData-Service with a top filter or a query filter, the Fabasoft OData does not know how to modify your query to include these both criteria.

That is why you have to add the placeholders

```
[TOPFILTER]
```

and

```
[CHANGEDATFILTER]
```

If we want to enable the \$top= option as well as the \$filter= option, we have to add both placeholders to our defined user query:

```
[TOPFILTER] SELECT * FROM FSCSCRUM@1.1001:Story WHERE [CHANGEDATFILTER] .objname LIKE "%Storage%" AND .objname LIKE "%Service%"
```

In the OData dialog it can look like that:

The screenshot shows the 'OData Service Model' dialog. At the top, there's a 'Type' dropdown set to 'Story'. Below it, the 'Properties' section has two buttons: 'Add Properties from "Story"' and 'Add Properties of a Form'. A table lists properties with checkboxes and their types:

Name	Type
<input type="checkbox"/> Name	String
<input type="checkbox"/> Sprint	Object
<input type="checkbox"/> Size (Story Points)	FSCSCRUM@1.1001:FibonacciList
<input type="checkbox"/> Fabasoft Cloud ID	String
<input type="checkbox"/> State	Object
<input type="checkbox"/> Feature	Object
<input type="checkbox"/> System Change Timestamp	Date/Time

Below the table are 'Add Entry' and 'Search and Add' buttons. The 'Provided Data' section shows 'User Query'. The 'User Query' field contains the following SQL query:

```
[TOPFILTER] SELECT * FROM FSCSCRUM@1.1001:Story WHERE [CHANGEDATFILTER] .objname LIKE "%Storage%" AND .objname LIKE "%Service%"
```

### 3.3 Filter Options

Fabasoft OData does support a subset of possible filters with the goal to optimize the performance. The supported filters are \$top (see chapter 8.3 'Performance Optimization "Getting Only X Rows"') and \$filter on the attribute objchangedat and its translated names. For the objchangedatfilter the attribute objchangedat (System\_Change\_Timestamp) has to be included in the attributes of an entity. Furthermore, filtering is only possible with the gteq (or ge) operator.

The syntax is:



/<entity>?\$filter=<objchangedat or translated name> ge|gteq <date and time with or without timezone>

**Note:** objchangedat would be the right name if your OData Service has “Use Programming name”. If not, it depends on the language of the OData Service. Means for an English OData Service the filter would look like filter=System\_Change\_Timestamp... and in German it would look like filter=System\_Änderungszeitpunkt...

An example query could look like this:

```
/Story?$filter=objchangedat gteq 2024-06-28T11:55:46
```

also possible with a time zone (be sure you encode the “+” with a “%2B”):

```
/Story?$filter=objchangedat gteq 2024-06-28T11:55:46%2B02:00
```

The \$filter and \$top filter can be combined by using &, for example:

```
/Story?$filter=objchangedat gteq 2024-05-28T11:55:46&$top=15
```

### 3.4 Other OData Parameters

OData knows a lot of other parameters. The Fabasoft OData supports only a subset of them:

- \$select  
Supported. You can e.g. define the attributes you want to get back like \$select=name,id
- \$top  
Supported (see chapter 8.3 ‘Performance Optimization “Getting Only X Rows”’).

\$filter

Supported (see chapter 0“

#### 3.4.1 User Query Together With Filter Options

If no user query is provided, a query will be created automatically. The filter options like

```
/Story?$filter=objchangedat gteq 2024-06-28T11:55:46
```

are then added automatically to the query.

But what if you provide a user query? It cannot be determined automatically, where in your user query to add the filter criteria.

**Example:**

You have a user query

```
SELECT * FROM FSCSCRUM@1.1001:Story WHERE .objname LIKE "%Storage%" AND .objname LIKE "%Service%"
```

If you want to request this OData-Service with a top filter or a query filter, the Fabasoft OData does not know how to modify your query to include these both criteria.

That is why you have to add the placeholders

[TOPFILTER]

and

[CHANGEDATFILTER]

If we want to enable the \$top= option as well as the \$filter= option, we have to add both placeholders to our defined user query:

```
[TOPFILTER] SELECT * FROM FSCSCRUM@1.1001:Story WHERE [CHANGEDATFILTER] .objname LIKE "%Storage%" AND .objname LIKE "%Service%"
```

In the OData dialog it can look like that:

The screenshot shows the 'OData Service Model' dialog. It has a 'Type' dropdown set to 'Story'. Below it is a 'Properties' section with two buttons: 'Add Properties from "Story"' and 'Add Properties of a Form'. A table lists properties with checkboxes and their types:

Name	Type
<input type="checkbox"/> Name	STR String
<input type="checkbox"/> Sprint	O Object
<input type="checkbox"/> Size (Story Points)	E FSCSCRUM@1.1001:FibonacciList
<input type="checkbox"/> Fabasoft Cloud ID	STR String
<input type="checkbox"/> State	O Object
<input type="checkbox"/> Feature	O Object
<input type="checkbox"/> System Change Timestamp	DT Date/Time

At the bottom of the properties section are 'Add Entry' and 'Search and Add' buttons. Below the properties is a 'Provided Data' section with a dropdown set to 'User Query'. At the very bottom is a 'User Query' section containing the SQL query: `[TOPFILTER] SELECT * FROM FSCSCRUM@1.1001:Story WHERE [CHANGEDATFILTER] .objname LIKE "%Storage%" AND .objname LIKE "%Service%"`. The query text is partially highlighted in red.

- Filter Options"). This is not affecting the performance as still all attributes are calculated. It only reduces the amount of data coming back.
- Other parameters like \$skip, \$expand, \$orderby and so on are **not** supported. They will be ignored.

### 3.5 Creating a Password for Applications

You need a password for applications to access your OData service. Create a password by opening the account menu (your user name) and clicking "Advanced Settings" > "Access for Applications". Create a password valid for "Open Data Protocol (OData)".

### 3.6 Licensing

OData licenses are available in 10M and 100M versions.

OData volume licenses are checked when

- the table overview for an OData service is retrieved.
- actual rows for a certain entity are retrieved.

The overview does not lower the number of available objects but the overview is not shown if there is no amount of any OData volume left.

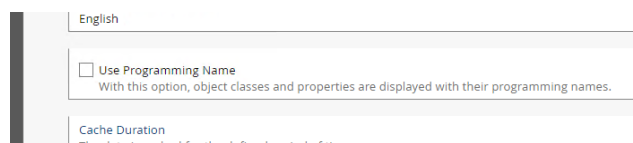
When retrieving rows for a certain entity the OData volume is lowered by the number of rows retrieved. That means you are charged exactly by the number of rows retrieved no matter which filters are applied and how many data rows there are in original.

## 4 Mapping of Fabasoft Cloud Objects to OData Entities

The input data are always objects in the Fabasoft Cloud. The OData service has the following strategy to map objects and attributes of objects to the OData entities and OData properties.

### 4.1 Mapping of Names

Depending if you have checked if you have checked the “Use Programming Names” option when creating an OData service, the names are generated differently.



#### 4.1.1 “Use Programming Names” Option

When “Use Programming Names” is checked, the internal reference names of Fabasoft Cloud objects and attributes are used as seen in this screenshot of the JSON output of the OData service:

```
1 {
2   "@odata.context": "https://at.cloud.fabasoft.com/odata/C00.6585.100.13.578666/$metadata#Story",
3   "value": [
4     {
5       "objaddress": "C00.6585.100.8.9141975",
6       "storystage": null,
7       "objsubject": null,
8       "storylead": null,
9       "storyrelease": null,
10      "storystakeholders": [],
11      "storystate": "C00.1.1001.1.219833",
12      "storysprint": "C00.6585.100.9.9891755",
13      "storyproject": "C00.6585.100.10.8828364"
14    },
15    {
16      "objaddress": "C00.6585.100.10.13343344"
```

If the option is not checked, the display names in the language select in the OData service is used.

The following rules apply:

- Special characters including whitespaces are replaced by an underscore.
- Leading and trailing whitespaces are not trimmed, the normal replacing logic applies.

```
1 {
2   "@odata.context": "https://at.cloud.fabasoft.com/odata/C00.6585.100.13.578666/$metadata#Story",
3   "value": [
4     {
5       "Fabasoft_Cloud_ID": "C00.6585.6000.3.8986843",
6       "Current_Stage": null,
7       "Subject": null,
8       "Lead": null,
9       "Release": null,
10      "Stakeholders": [],
11      "State": "C00.1.1001.1.219833",
12      "Sprint": null,
13      "Scrum_Project": "C00.6585.100.9.8766663"
14    },
15    {
16      "objaddress": "C00.6585.100.10.13343344"
```

## 4.2 Attribute Value Generation

Values of attributes in the Fabasoft Cloud are mapped into a value in OData. Here the following rules apply:

### 4.2.1 Basic Types

- Basic data types are mapped directly. Integer types get an integer type in OData. String types get a string type in OData.
- The only exception here is the basic type "Currency" as this is also an aggregate where special rules apply (see next point).

### 4.2.2 Lists

- Lists are mapped directly into OData/JSON lists.

### 4.2.3 Aggregates

- Aggregates are flattened out. That happens by concatenating the aggregate name with the attribute name with an underscore as separator.

So, if you have an aggregate named `Lock Information` and there inside the attributes `Is Locked`, `Locked By` and `Locked Since` it will look like that:

```
1 {
2   "@odata.context": "https://at.cloud.fabasoft.com/odata/C00.6505.1.8.1
3   "value": [
4     {
5       "Lock_Information_Is_Locked": "true",
6       "Lock_Information_Locked_By": "C00.6505.100.10.8151285",
7       "Lock_Information_Locked_Since": "2023-07-04T14:08:40+01:00"
8     }
9   ]
10 }
```

Important is, that these 3 properties are only here, if the `Lock Information` aggregate is set at all. So, if it is null, these 3 properties won't be in the result set.

- Currencies are aggregates as well. They are flattened out with the two options `currvalue` and `currsymbol` so if you have a currency data type on an attribute `Price` it could look like that:

```
1 {
2   "@odata.context": "https://at.cloud.fabasoft
3   "value": [
4     {
5       "Price_currvalue": "94.06000000000000",
6       "Price_currsymbol": "Euro"
7     }
8   ]
9 }
```

## 5 In-Depth Explanation: Execution of OData Requests

There are three main types of requests possible:

1. OData GET request to an OData Service
2. OData GET request to an entity of an OData Service
3. OData precalc (POST or GET)

## 5.1 OData GET Request to an OData Service

When OData GET requests are done, the OData Service in the Fabasoft Cloud is loaded and the needed information about the OData schema are calculated. The result is a valid OData overview.

## 5.2 OData GET Request to an Entity of an OData Service

On such a request, actually data is loading from the Fabasoft Cloud and given back in OData style.

The whole process can be split up in these parts:

1. Generating the schema.
2. Calculating the cube.
3. Getting the attribute values.
4. Building the OData response.

Especially the second and the third step can be performance critical, so these are described in more detail.

### 5.2.1 Calculating the Cube

Given an entity defined in the OData Service as “Found Objects”, two queries to the Fabasoft Cloud are done to fetch the data:

First a query is done which looks up all Teamrooms in the given context (room, configuration or organization). This list of Teamrooms is then used in the second query, which looks up objects of the entity type which are in the given list of Teamrooms.

**Important:** Please be careful here about limits: if your selected context (room, configuration or organization) has a lot of Teamrooms (10 000+), this can cause the second query which is an IN-Query, to not run performant or not run at all because of too many IN-values. Please consider here defining your own query for this entity!

### 5.2.2 Getting the Attribute Values

In parallel running threads/processes, the wanted attributes are loaded. On performance issues please check out the performance chapter.

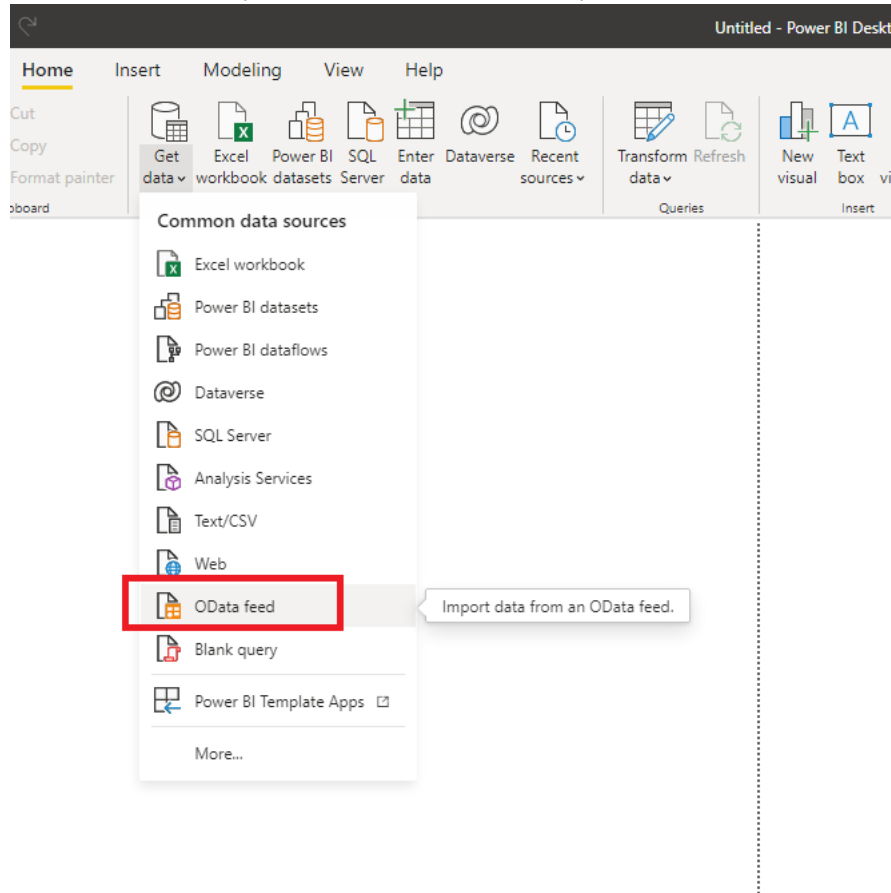
## 6 Access the OData Service via Microsoft Power BI

Using Microsoft Power BI to access the data provided by the OData service is quick and easy. Microsoft Power BI is a great tool for creating reports and statistics.

For more details on how to work with Microsoft Power BI please refer to:  
<https://powerbi.microsoft.com/en-us/support/>

To add the OData service to Microsoft Power BI, proceed as follows:

1. On the “Home” tab, click the “Get data” button, and then click “OData feed” as data source.



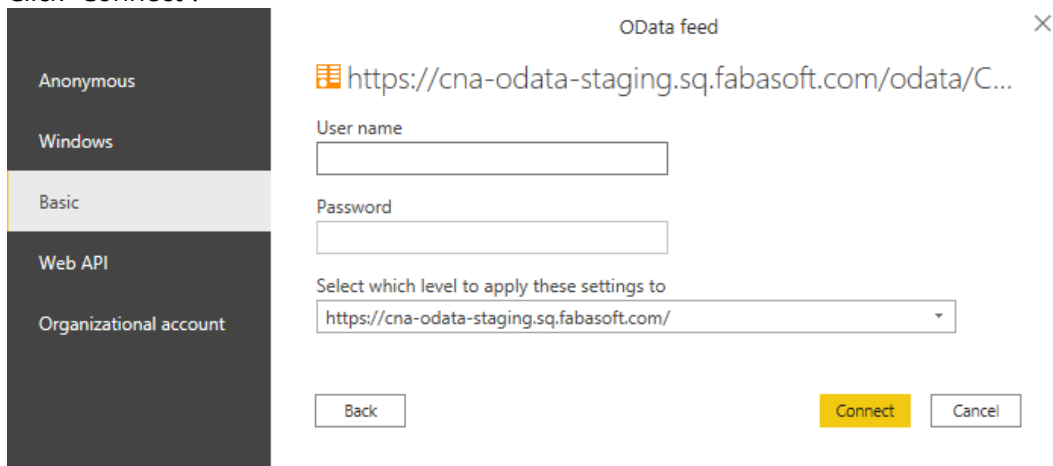
2. Enter the URL of the OData service (see chapter 3.1 “Creating an OData Service”) in the *URL* field and click “OK”.

**Note:** The “Basic” connection option is sufficient.

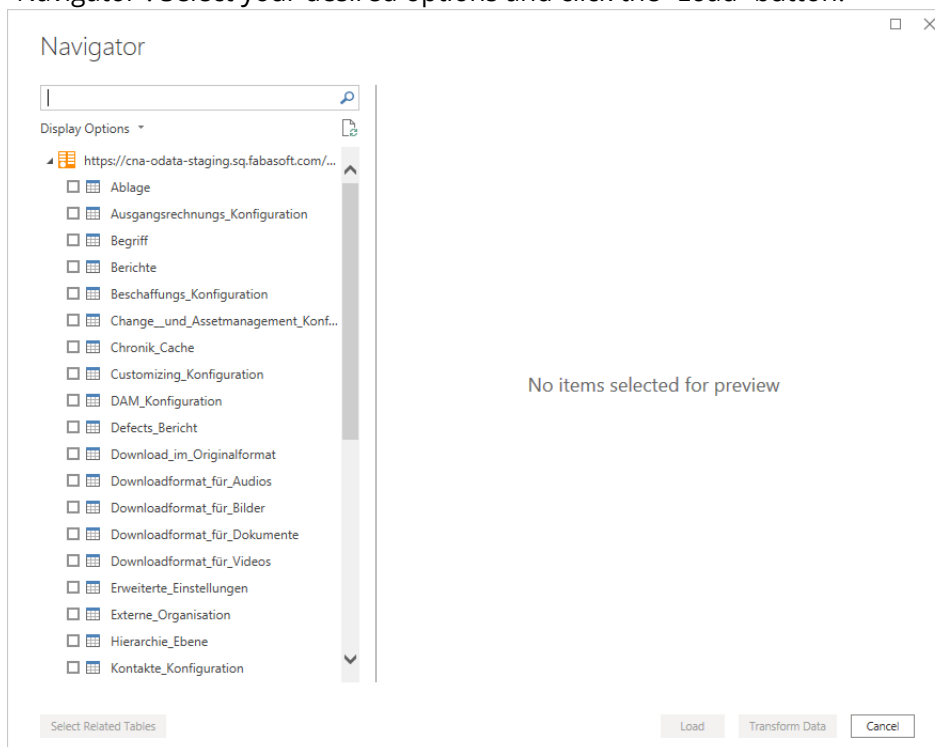


Since OData usage requires authentication you need to select the “Basic” authentication option. Enter your e-mail address for the cloud log-in as the user name and the generated Password for Applications as the password (see chapter 3.5 “Creating a Password for Applications”).

3. Click “Connect”.



4. After successful authentication, you can select from several display options in the “Navigator”. Select your desired options and click the “Load” button.

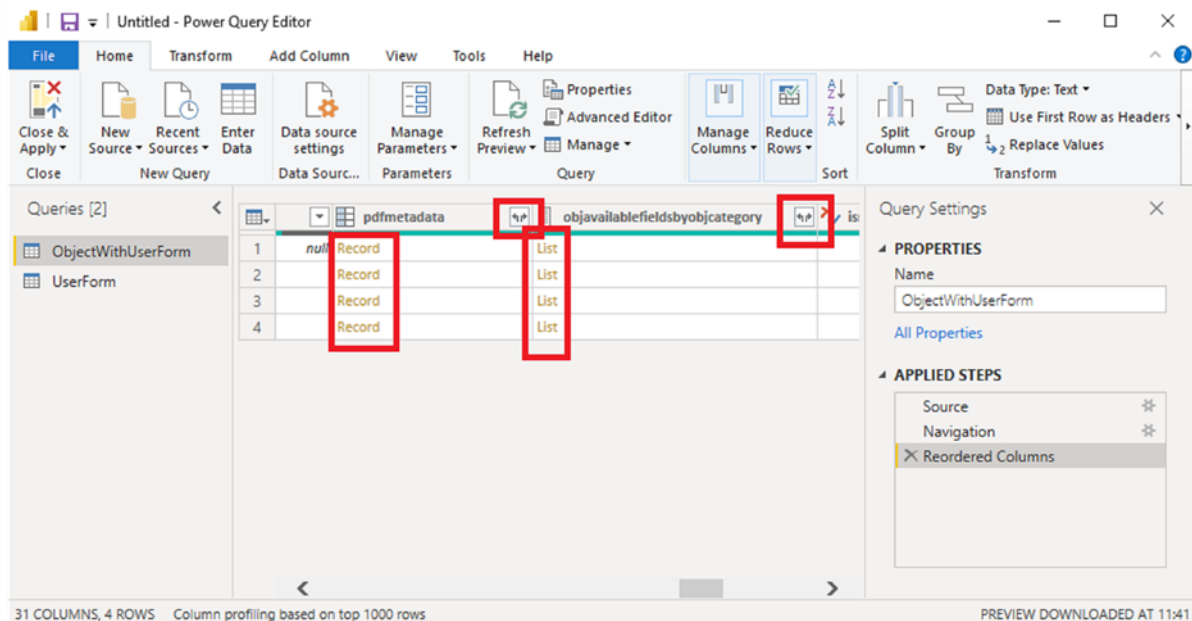


## Lists

When you create an OData service, it is very likely that some of the required properties have lists as data types.

Lists are displayed as “List”. In the “Transform Data” dialog you can see these lists.

In order to make them useable, click on the “Expand” tool at the end of the column (see red box on the column in the screenshot) to choose an expanding option. Such an option is e.g. creating a new table; however, this requires additional work as relationships between the tables need to be set.



For lists, the approach to make them useable depends on the expected number of entries. If you can predict having only a couple of entries, expanding the list is an option. In this case, the other values are duplicated. If there are many entries, creating a new table based on the list may be a better solution.

## 7 Access the OData Service as a Developer

The OData protocol is a standardized HTTP-based protocol to retrieve data. Please check <https://www.odata.org/> for details regarding the standard, programming examples and more.

When accessing an OData service URL via a web browser, you can interactively see the results of OData queries as JSON documents or XML documents.

Accessing data via an OData service is quite simple. Basically, all you need is to create an HTTP-based request with basic authentication turned on. The user name is the e-mail address and the password is the one created via "Password for Applications".

The response is always a JSON document. In many programming languages, there is already a good toolset to parse and interpret JSON.

**Note:** If you have to process the metadata as well, simply follow the URLs stored in `@odata.context` which you can see in the screenshot below.



```

{
  "@odata.context": "http://[redacted]/odata/C00.1.506.2.3542/$metadata#ObjectWithUserForm",
  "value": [
    {
      "objaddress": "C00.1.506.3.3542",
      "objclass": "C00.1.1001.1.364527",
      "objname": "sadi",
      "objcreatedby": "C00.1.506.1.5150",
      "objcreatedat": "2021-07-21T10:53:43+02:00",
      "objchangedby": "C00.1.506.1.5150",
      "objmodifiedat": "2021-07-21T10:53:48+02:00",
      "objsecchangedat": "2021-07-21T10:53:48+02:00",
      "objchangedat": "2021-07-21T10:53:48+02:00",
      "objuserchangedat": "2021-07-21T10:53:48+02:00",
      "objowner": "C00.1.506.1.5899",
      "objowngroup": "C00.1.506.1.5897",
      "objaclobj": "C00.1.1001.1.331843",
      "objactversnr": 1,
      "objactverscreated": "2021-07-21T10:53:43+02:00",
    }
  ]
}

```

## 8 Performance

The Fabasoft Integration for OData processes data in of two steps:

- During the first step ("Cube creation"), all objects as defined in the data source are queried and loaded into the OData service. The scope of objects considered is defined by the context of the data source (e.g. Teamroom). If the objects of the query result contain references to instances of classes contained in the data source, these instances are added to the object set. No queries are executed for the object classes "Object", "User" and "Group". Instead, the references to these instances are collected in the corresponding tables.
- During the second step ("Table creation"), the values per table are loaded and mapped to the structure as required by the OData query processor. This data is created per table on demand. However, all of the table rows are created independent of the amount of table rows queried.

Keep in mind that any caching within the OData services is performed per user. Thus, if multiple users access an OData service, they all work with their dedicated OData cache.

### 8.1 Performance of the Initial Call to an OData Service

In order to reduce the processing overhead of the initial call to an OData service you can

- reduce the number of entities or tables, and/or
- reduce the number of properties inside entities or tables with high row count.  
Try removing unused properties to reduce the overall processing time for each entity. This is most likely the fastest way to resolve such performance issues.

### 8.2 Performance of Subsequent Calls to OData Service

Should the OData service lack performance with regards to subsequent calls, you may increase the value in the *Cache Duration* field. In this case, the data is stored in the cache for a longer period, hence subsequent calls may reuse the already existing data table.

### 8.3 Performance Optimization “Getting Only X Rows”

In OData request syntax there is a filter targeting the number of rows fetched. It is the “top” command and can be used as followed:

```
odata/COO.1.506.4.2141/Scrum_Story?$top=500
```

This top command is also used in some business intelligence tools like Microsoft Power BI when showing a preview of the data.

Because it is commonly used this command has been optimized. That means internally only that amount of entities are fetched and processed. For the user it is transparent – it just does what it should do, just a little faster than other commands.

### 8.4 Precalculation

If reducing the number of attributes as well as splitting up to less entities per OData service does not help to get the result within the given time window (30 minutes), you can use precalculation.

Precalculation is triggered similar like calling an OData service, just with `precalc` in the URL and a HTTP POST instead of HTTP GET.

The precalculation is performed asynchronously on the server side. That means, when you trigger the precalculation for an OData service, you will quickly get back a 200 OK. In the background it will precalculate the OData service by writing a cache into a Teamroom in the Fabasoft Cloud.

To enable the precalculation, two steps are needed in the OData service:

- Ensure a valid cache duration is entered. 24 hours is our suggestion. That means, you could trigger the precalculation at night and then on the whole day these cached entries are taken to quickly deliver OData results.
- A Teamroom has to be entered below where the cached data is stored. Inside this Teamroom, another Teamroom will be created especially for you. Please be careful with giving additional permissions on these inner Teamroom to others as it may be that others can see data they normally have no access to.

☐ Use programming name  
With this option, object classes and properties are displayed

☐ Flatten Out Aggregates  
Flatten out aggregates for performance. In the case of nest

**Cache Duration**  
The data is cached for the defined period of time.  
24 h      min

**Teamroom**  
The Teamroom used for storing the cache.  
☒ Cache for OData Teamroom

**Restrict Data to Objects From \***  
Define the scope of the OData service. Only data from objects  
Neufassung von Scrum-Projekt

Finally, you have to call the precalculation. If the normal URL for your OData service is

```
GET https://at.cloud.fabasoft.com/odata/precalc/COO.200.100.15.301
```

you have to add `/precalc` after `/odata` and use the HTTP POST method instead of HTTP GET.

This could be your final URL:

`https://at.cloud.fabasoft.com/odata/precalc/COO.200.100.15.301`

You can use e.g. `curl` or Postman to trigger that request. The authentication is the same as on a regular OData request – basic authentication with your username and the OData password from password for applications.

The screenshot shows the Postman interface for a POST request. The URL bar contains `https://at.cloud.fabasoft.com/odata/precalc/COO.200.100.15.301`. The 'Authorization' tab is selected, showing 'Basic Auth' as the type. The 'Username' field contains `first.surename@fabasoft.com` and the 'Password' field is masked with dots. A 'Show Password' checkbox is visible below the password field. A note states: 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'.

Depending on your OData service and your data, it can take hours to be finished. Triggering it at night time makes sense.

The actual GET OData request should then be way quicker compared to without preloading. Depending on the amount of returned data entries, it can then take seconds to minutes to be finished.

## 9 Restrictions

The following restrictions apply.

### 9.1 Data Types

There is no support for content or dictionary properties.

### 9.2 Categories and Entities

Each entity can only be used once in an OData service.

In the following working example, you can see as entity a story which only appears if an "IT-Einkaufsvertrag" is chosen as category.

The screenshot shows a 'Model' dialog box with an 'Add Entry' button. Below the button is a table with two columns: an unchecked checkbox and 'Typ'. The table contains two rows: '1 Object' and '2 IT-Einkaufsvertrag/Story'.

	Typ
<input type="checkbox"/>	
1	Object
2	IT-Einkaufsvertrag/Story

But this story **cannot** appear twice in the model:

Model

Add Entry

	Typ
1	Object
2	IT-Einkaufsvertrag Story
3	Story

### 9.3 User Forms

Only the following use cases with user forms are supported:

- Adding properties of an existing user form.
- User form data types: primitive data types, lists of strings.
- Adding properties of more than one user form.