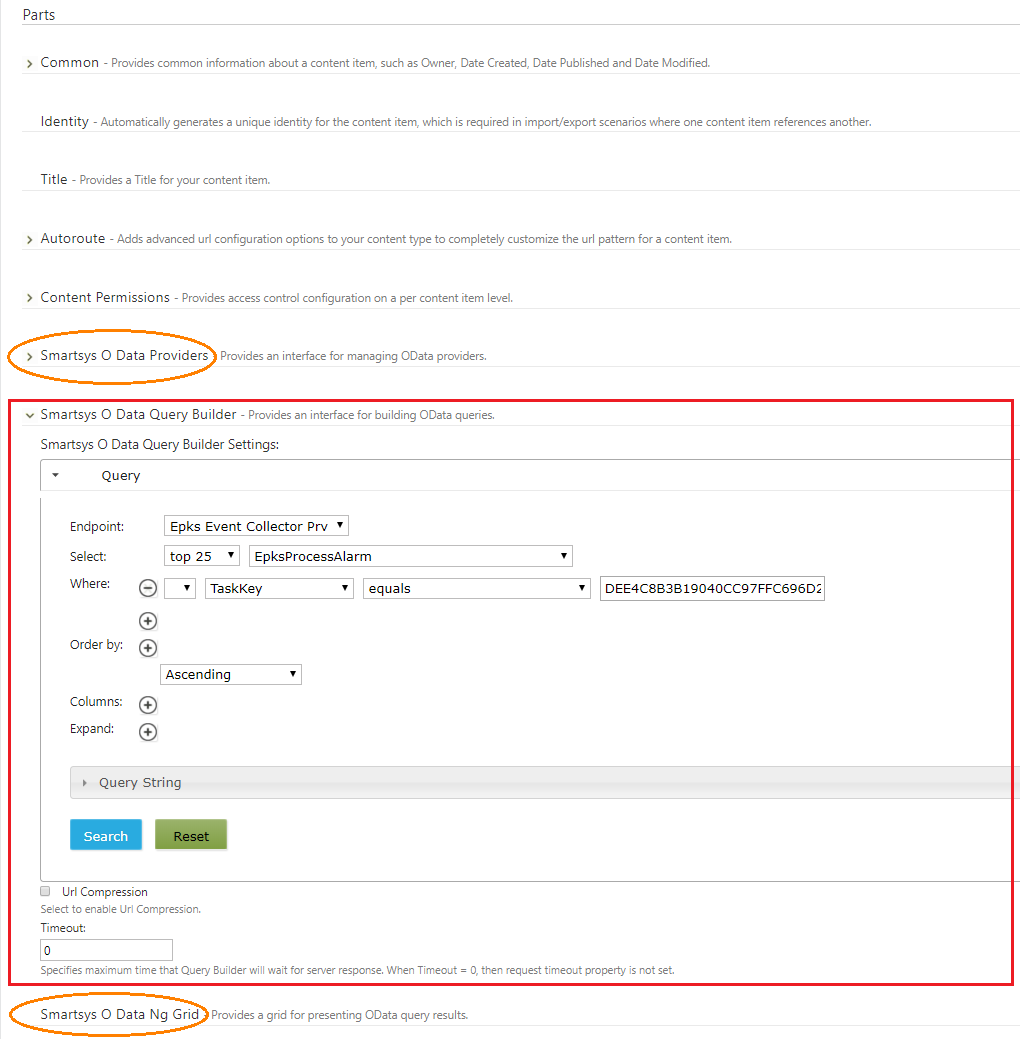
This chapter describes how to create WEB pages with built-in functionality for the [OData](https://www.odata.org/) queries. The Smartsys OData Query Builder module provides a Content Part, with the same name that may be attached to every Content Type. This provides a user interface for building [OData](https://www.odata.org/) queries, as well as, a tool for constructing queries without the need for prior knowledge of [OData's syntax and rules](https://www.odata.org/documentation/odata-version-2-0/uri-conventions/).

## Query Builder in Content Types

Query Builder, in the context of the Content Types configuration, gives an option for pre-configuration of all query properties discussed later in this chapter. Values that are configured here will be inherited as defaultsettings by newly created Content Items.

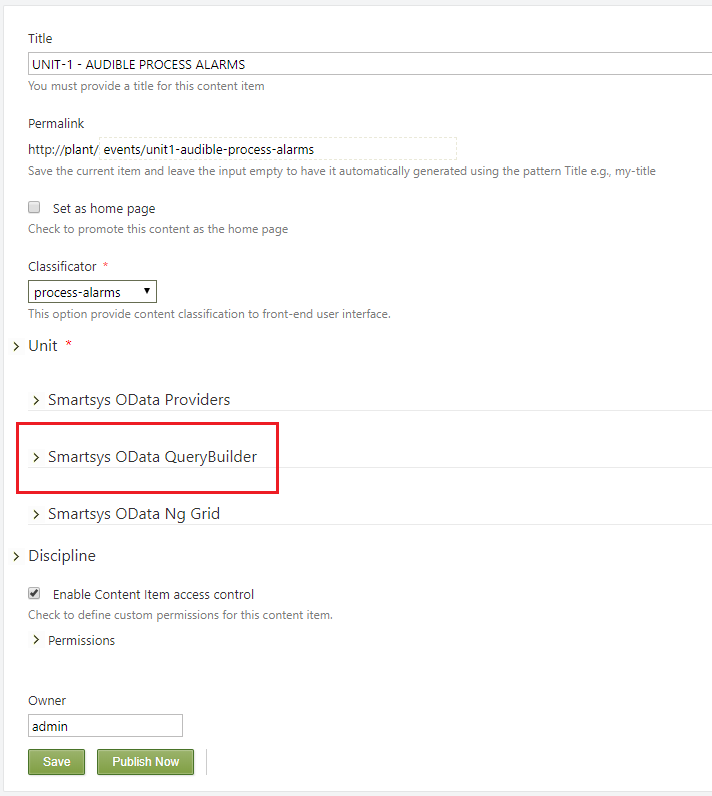


!!!note "Note:" Note that besides the OData Query Builder, there are two other essential content parts configured in this content type - OData Providers and OData Ng Grid. Their role is to provide OData endpoints and skin for data presentation. For more details, see chapters [Access to Relational Data](file:///C:\access-to-relational-data) and [OData Ng Grid](file:///C:\odata-ng-grid). General knowledge about content type concepts is available in chapter [Content Anatomy](file:///C:\content-anatomy).

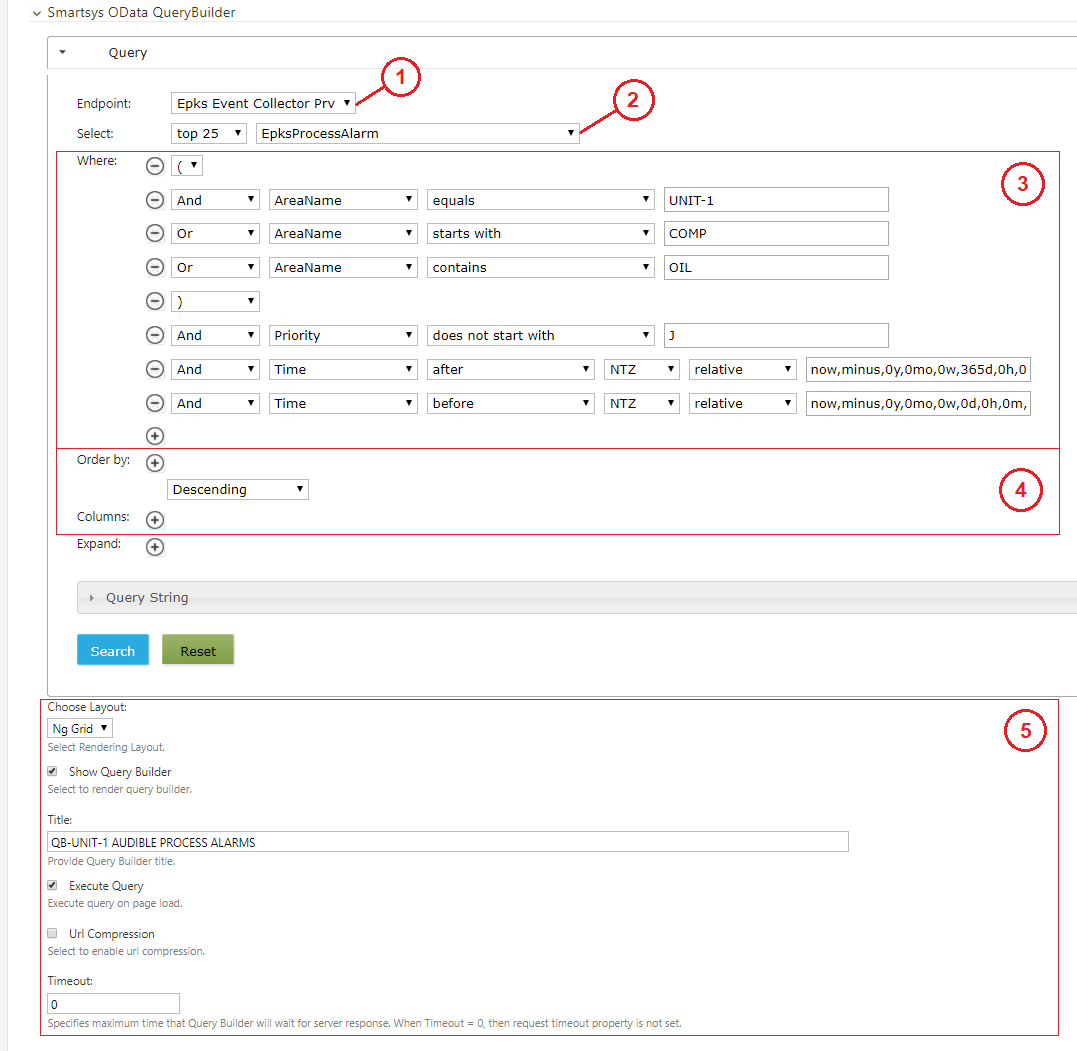
## Query Builder in Content Items

Query builder, in the context of Content Items, is available, when creating or editing a content item from a content Type, where the Query Builder part is present.

The following image presents the look of the Content Item configuration:



To begin the configuration, open the OData Query Builder configuration container, then open the Query accordion, as shown on the next screenshot.



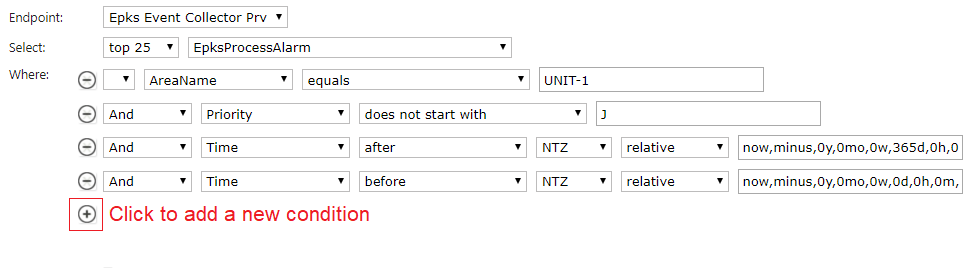
The configuration steps are as follows:

(1) - Choose an endpoint. You can customize which endpoint you want to use. The OData Providers part allows for selection between multiple services.  
(2) - Select the entity that you want to query, as well as, the number of results needed. Every endpoint service provides one or more entities. The entity often corresponds to the relational database table.  
(3) - Configure the search conditions according to the information that is needed to be displayed for the given content item. The system provides an intuitive interface for building relatively complex where clauses.  
(4) - Sort your results and choose the columns you want to be present on the front-end interface.  
(5) - Additionally, you can setup the following query properties, usually preconfigured during content type configuration:

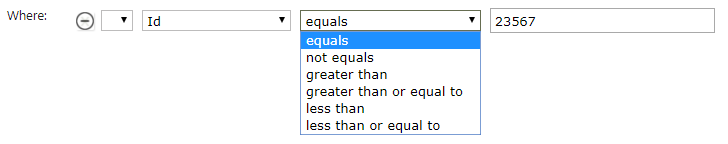
* Choose Layout: - Select one of the available skins for front-end data presentation.
* Show Query Builder: - By activating this checkbox, you will instruct the system to render query builder on the front-end page, thus allowing the users to change the default query. The user can set up their queries and execute them against the endpoint. It is handy when the administrator prepares the query, and the user has to change a piece and get a new result.
* Title: - You can provide a title for the query builder. The title will be visible on the front-end and act as a link and toggle open/close state of the builder.
* Execute Query: - If checkbox is select, then the predefined query is executing immediately` after page load, and the result is displaying. Otherwise, query execution shall be trigger by the user or another page component.
* Url Compression: - The selected checkbox will instruct query builder to compress the Url request before submitting it to the server.
* Timeout: - This value specifies the maximum time to wait for a server response before displaying an error message. If the value is set to **zero**, then the timeout is not monitoring.

## OData Query Stepwise

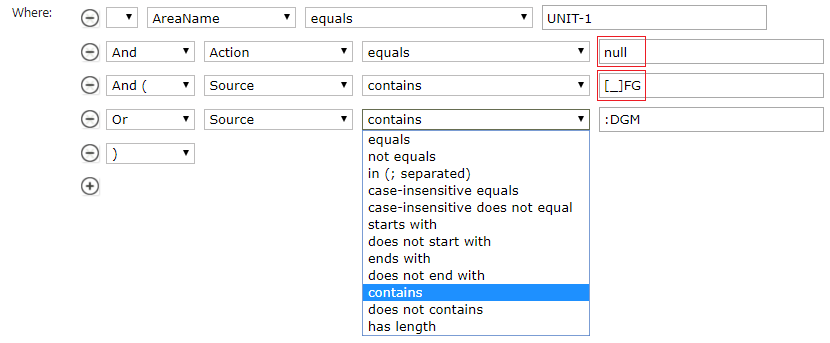
Once you configure the endpoint and entity fields, the next step is to build a conditional Where Clause to filter out the results. The where Clause can be combined with AND, OR and NOT logic operators, as well as, brackets to control execution priority. To add a new row in the condition, click on the (+), shown in the image below, and then select a logic operator from the first drop-down selector.



The query builder uses entity metadata to discover entity columns and [primitive datatypes](https://www.odata.org/documentation/odata-version-2-0/overview/). The second drop-down selector lists all available columns, where you can select one for the current logic row. Depending on the datatype, the third drop-down automatically proposes possible **condition operators** for the given datatype. In the last field, you have to provide an **operand** value according to your selected requirements. The following example uses a column with the name Id, which has a datatype **Integer**. The query builder automatically proposes allowed logic operators for this datatype.

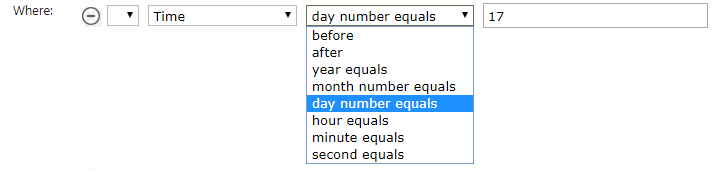


Another frequently used datatype is the **string** datatype. The next snippet adds a condition for selecting data, where the column Action has a NULL value string, and the column Source contains one of the substrings \_FG or :DGM. The drop-down list shows operators that are available for the string datatype.

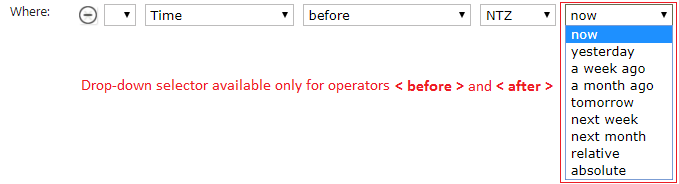


!!!attention "Attention" When the condition includes the selection of records, where the string has a NULL value, use null as an operand value. When the operand substring contains an underscore character use squire brackets to enclose the character [\_].

In the case where the operand column has a **date-time** datatype, a more interesting scenario occurs. The following image shows the available operators for this datatype:



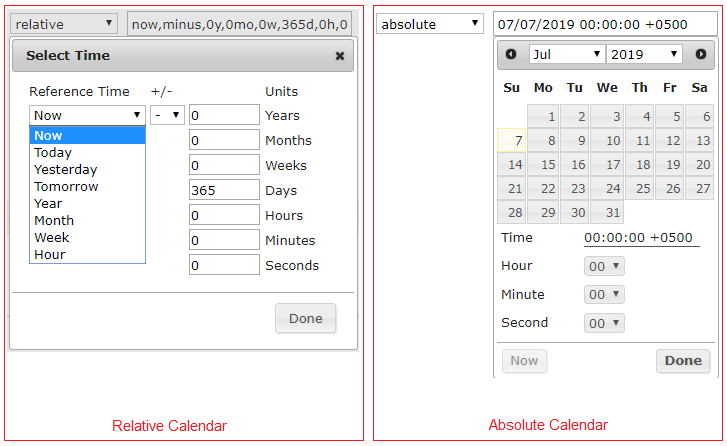
With the exception of the first two operators, the configuration of the rest is very similar to any other datatype. It consists of the selection of a logical operator and providing an operand value. When you select before or after, then the interface automatically shows additional drop-down selectors.



The first options in the list are relative time aliases. For example, if **now** the time is 07/07/2019 15:20:16. The time aliases will point to the following times:

* yesterday - 06/07/2019 15:20:16
* a week ago - 06/30/2019 15:20:16
* tomorrow - 07/08/2019 15:20:16

The last two options provide a way for precise relative time selection and absolute time selection. When one of these options is selected, the interface shows an additional box on the right. Clicking on that box will display the relative or absolute calendar.

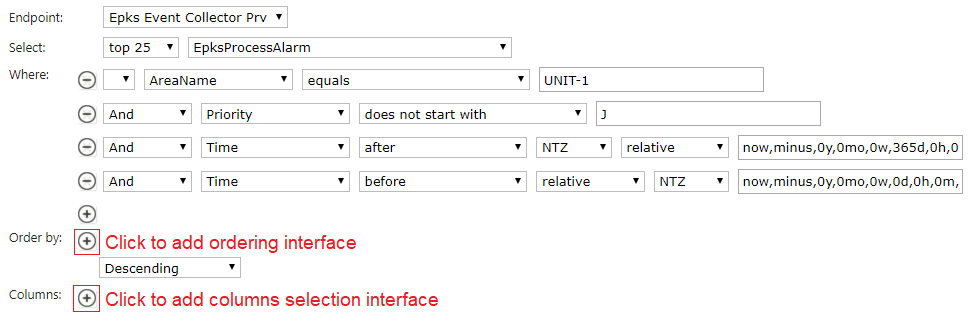


A reference time in the relative calendar provides a list with time aliases. For example, if the current time is 06/28/2019 15:20:16, then the meaning of the time aliases would be:

* Now - 06/28/2019 15:20:16
* Today - 06/28/2019 00:00:00
* Yesterday - 06/27/2019 00:00:00
* Tomorrow - 06/28/2019 00: 00: 00
* Year - 01/01/2019 00: 00: 00
* Month - 06/01/2019 00: 00: 00
* Week - 06/24/2019 00: 00: 00
* Hour - 06/24/2019 15: 00: 00

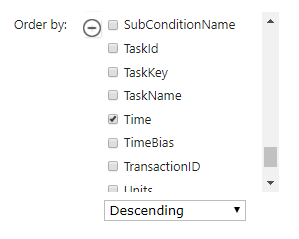
Additionally, you can add a +/- time offset by entering the required figures in the corresponding cells from the right column.

To sort the results and prepare the data for front-end presentation - click on the (+) in the Ordered By section. This will add an interface for data ordering. In the Columns section, click on the (+) to add an interface for column selection.

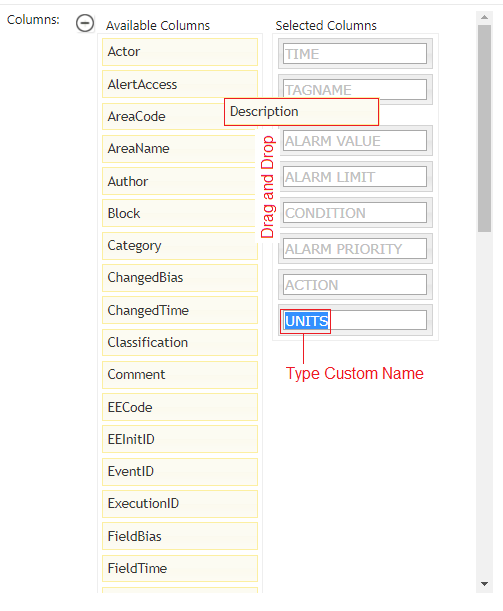


The query builder lists all available entity columns in Ordering by and Columns interface areas.

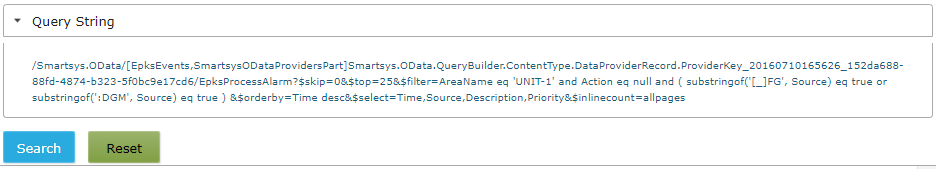
By selecting the checkboxes in the Ordering by area, you instruct the server to sort the results by the data coming from the corresponding columns. Multiple sorting criteria are available, they will execute in the same order in which they are selected during configuration. You can also choose between Ascending and Descending order criteria from the drop-down selector on the bottom of Ordering by area. In the next example, the returned data is sorted by **Time** in descending order.



In the Columns area, all columns from the current entity are available on the left side. You can move the columns by drag and drop to the container on the right side under the title Selected Columns. To remove the column from selected columns, drag and drop them from the right side to the left side. The columns on the right appear with their original names. You can rename them by double clicking on the cell. You can also order the selected columns within the container by vertical drag and drop. Data presentation skin will collect this configuration information and render only the selected columns in the same order to the front-end.



At the bottom of the query builder area, you will see a part of an accordion with the name Query String, as well as, the Search and Reset buttons. Inside the accordion area, a Url request that is automatically prepared by the query builder is ready to be sent to the endpoint. You can even execute this OData query in the browser and obtain raw data. This data is an input for the presentation skin that wraps it and then presents the final layout to the users.



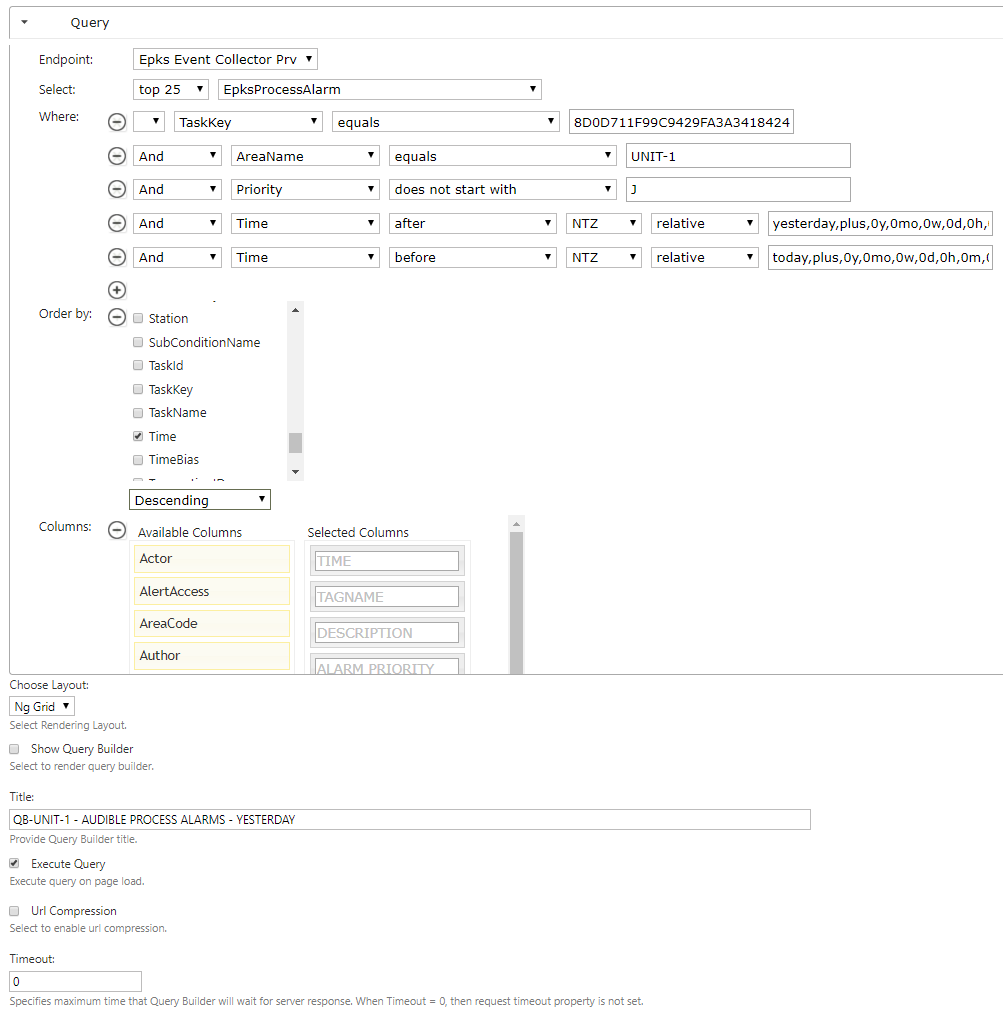
The Search button is for testing purposes of the query from the back-end. The button will execute the query and then render the results using simple skin below the buttons. The Reset button will clear all current query configurations.

## OData Query Archetype

Now let’s take a look at an example.

!!!tip "The Mission:"  
We have to create a content item that presents the results for annunciated process alarms generated by tags with the parent asset UNIT-1. The page has to be able to query the events for the last day, and list a maximum of 25 results. The results must be sorted by timestamps in descending order. The users must be able to see: TIME, TAGNAME, DESCRIPTOR and ALARM PRIORITY. The users must not be able to change the query.

The following image shows OData query configuration that complies with the task description.



!!!note "Note:" It's good practice to add a column included in the entity index in the search criteria when dealing with large databases. This can improve endpoint performance. In the example above - TaskKey is included in the index of EpksProcessAlarm entity. The [Access to Relational Data](file:///C:\access-to-relational-data) chapter explains TaskKeys.

In this chapter, we have learned how to configure the OData Query Builder in the contexts of Content Types and Content Items. To gain a complete understanding of the entire process of setting up endpoints and rendering the results to the front-end, read the chapters [Access to Relational Data](file:///C:\access-to-relational-data) and [OData Ng Grid](file:///C:\odata-ng-grid). The [Content Anatomy](file:///C:\content-anatomy) chapter contains a more detailed view of Content Type and Content Item concepts.