

The **Smart search filter** [web part](#) allows users to limit the scope of the data that will be searched (conditional filter), or define the order of the search results. You can connect any number of filters to the following types of search web parts:

- **Smart search dialog**
- **Smart search dialog with results**
- **Smart search results**




Note: The *Smart search filter* web part only work with [locally stored indexes](#). When using [Azure Search indexes](#), you can create custom components for search filtering. See [Integrating Azure Search into pages](#) for more information.

The behavior of the smart search filter is primarily defined by the web part properties described below.



You can find information about the other properties by clicking the help link in the corner of the web part properties dialog.

Property	Description
Search web part ID	Enter the Web part control ID of the <i>Smart search dialog</i> , <i>Smart search dialog with results</i> , or <i>Smart search results</i> web part that you wish to connect to the filter. The search web part must be on the same page as the filter.
Filter mode	Sets the user interface type of the filter. Possible choices are: <ul style="list-style-type: none"> • Drop-down list • Checkboxes • Radio buttons • Text box
Filter auto postback	Indicates whether the search results automatically refresh (via postback) whenever a user selects a different filtering option. Not applicable when using the <i>Text box</i> Filter mode . If you enable auto postback for a search filter, also set the Paging mode property to Postback for the related <i>Smart search dialog with results</i> or <i>Smart search results</i> web part.
Values	Determines the available filtering options. See the Defining filtering options section below for details.
Query name	Allows you to generate the filtering options dynamically based on data in the system. Specify the name of an SQL query that loads the required data (instead of using the Values property). Enter the full code name of the query or Select an existing query. The query must return the appropriate values depending on the type of the filter. For example, for a standard conditional filter, the query needs to return three columns in the following order: <i><index field name></i> , <i><value of the field></i> , <i><displayed text></i> For more information, see Example - Loading filtering options dynamically .

Filter clause	<p>Sets the logical value for the entire conditional filtering clause generated by the filtering options. Possible choices are:</p> <ul style="list-style-type: none"> • None – no clause is added and the original logical values set for individual filtering options are used. • Must – indicates that the conditions of filtering options must be fulfilled. • Must not – indicates that the conditions of filtering options must not be fulfilled. Conditions are inverted compared to the <i>Must</i> option. <div style="background-color: #e1f5fe; padding: 10px; margin-top: 10px;"> <p> Tip: When using <i>Checkboxes</i> filtering mode, you can create filters that display search results fulfilling at least one of the selected filtering options:</p> <ol style="list-style-type: none"> 1. Remove any logical clauses (+ or - symbols) from individual filtering options in the Values property. 2. Set the Filter clause to <i>Must</i>. <p>If you set the Filter clause to <i>None</i> and add the + symbol to individual option values, the filter displays only search results that fulfill all of the selected filtering options.</p> </div>
Filter is conditional	If true, the filter limits the scope of the objects that are searched (where condition). If false, the filter determines the order in which the search results are displayed (order by condition).

You can find examples of search filters on the sample Corporate Site on the **Examples -> Web parts -> Full-text search -> Smart search -> Smart search filter** and **Faceted search** pages.

Defining filtering options

The most important part of configuring Smart search filter web parts is the definition of the filtering options offered to users. In most cases, you will set the options through the **Values** property of the web part. If you wish to use the **Query name** property to load the options dynamically, the rules described below also apply to the results retrieved by the query.

The format of the filtering options depends on the type of the filter.

Conditional filters

In the case of conditional filters, define one option per line in format:

Index field name;Value of the field;Displayed text

You need to specify the logical meaning of each filtering option by adding the + or - symbol as a prefix:

+	The search only returns objects whose value in the field matches the value specified in the second part of the filtering option's definition.
-	The search excludes all results whose value in the field matches the value specified in the second part of the filtering option's definition.

To create filtering options that check the values of multiple index fields, use the following syntax:

(Field1:(Value1) OR Field2:(Value2));;Text

If you wish to use the same value for all fields in the condition clause, you can insert the value using the **{0}** expression:

(Field1{0} OR Field2{0});Value;Text

You can also use the OR operator in the value section of filtering options to check whether a field matches one of multiple possible values. Syntax:

Field;Value1 OR Value2 OR Value3;Text



When entering **integer**, **floating-point(double)** or **decimal** type fields as filter options, you need to specify the type of the value:

`+DocumentCreatedByUserID;(int)53;Administrator`

Examples:

- `;;All`
- `+classname;cms.smartphone;Smartphones`
- `+_created;[{%ToSearchDateTime(CurrentDateTime.AddDays(-7))}%} TO {%ToSearchDateTime(CurrentDateTime)%}]; Past week`
- `+_content;product;Results related to products`
- `+skudepartmentid;(int)4 OR (int) 6;Products in specific departments`
- `+(documenttags{0} OR _content{0});product;Results related to products`
- `-(issecurednode:(true) OR requiressl:((int)1));Exclude secured pages`

Search result order filters

When creating filters that change the order of the results (i.e. the **Filter is conditional** property is disabled), define options in the following format:

Index field name;;Displayed text

The order is determined according to the values in the specified field.



Notes:

- By default, the order is ascending. To make the order descending, add the DESC keyword after the field name.
- Use the `##SCORE##` macro instead of the field name to order results according to their search relevance (the score order is always descending and cannot be reversed).

Examples:

- `##SCORE##;;Score`
- `documentcreatedwhen;;Creation date`
- `SKUPrice DESC;;Price descending`

Text box filter mode


Text box filters do not offer any selection options, so the **Values** property instead determines which index fields the system searches when a user enters an expression into the text box. You can specify multiple fields, each one on a new line. Like with standard conditional filters, the + and - symbols determine whether the search returns results that contain the text box value in the given field, or only those that do not.

For example, if a text box filter has `+DocumentTags` in the **Values** definition, visitors can enter the names of [page tags](#) into the text box and perform a standard search. The filter modifies the retrieved results to contain only pages that are marked by the specified tags.


Use the following syntax to create OR conditions that are fulfilled if the text box value matches at least one of multiple fields: ***(Field1{0} OR Field2{0})***

For example: `+(DocumentTags{0} OR _content{0})`

The `{0}` expression represents the value that users type into the text box filter. The search internally converts the expression to `Field:(value)` in the resulting condition.

 **Tip:** To create a text box filter that behaves like a regular search box, use `+_content` as the only field name.

You can also use text box filters to determine the order of the search results. In this case, disable the filter web part's **Filter is conditional** property and leave the **Values** property empty. Users can then enter the name of the field used for ordering into the text box. This scenario is not recommended for use by regular visitors on the live site.

 In order for the filter to work correctly, the index fields used in the option definitions must have the **Searchable** option enabled in the **Local** search field configuration of the given object type.

You can also create filtering options for the fields that are marked as **Content** by using `+_content` as the field name. Such conditions are fulfilled if the value is found in any of the content fields.


Example - Loading filtering options dynamically

The examples in this section demonstrate how to create conditional filters with dynamically loaded options. The sample filters are intended for product search results.

The first filter provides checkboxes representing all available product page types. Add a **Smart search filter** web part to your search results page and configure the following properties:

- **Filter mode:** Checkbox
- **Query name:** Create a **New** query with the following code:

```
SELECT 'classname', ClassName, ClassDisplayName
FROM CMS_Class
WHERE ##WHERE##
ORDER BY ##ORDERBY##
```

 The `##WHERE##` and `##ORDERBY##` expressions are placeholders that are replaced by the values of the web part's **Query WHERE condition** and **Query ORDER BY clause** properties.

- **Query WHERE condition:** `ClassIsDocumentType = 1 AND ClassIsProduct = 1`
- **Query ORDER BY clause:** `ClassDisplayName`
- **Filter clause:** Must
- **Filter is conditional:** Yes (selected)

The query returns three columns:

- **'classname'** – a static value that is the same for all filtering options. Specifies the name of the search index field whose value the condition checks.
- **ClassName** – the value of the `ClassName` column for records of the `CMS_Class` table. Search results must match this value to fulfill the filter's condition when the given option is selected.
- **ClassDisplayName** – the value of the `ClassDisplayName` column for records of the `CMS_Class` table. Used for the text captions of the checkboxes in the filter.

The following are examples of rows that the query could return:

classname	DancingGoat.Coffee	Coffee
classname	DancingGoat.Ebook	E-book
classname	DancingGoat.Kettle	Kettle

Setting the **Filter clause** property to **Must** means that the *classname* value of search results must match the value of at least one of the selected checkbox options to fulfill the filter's condition.

The web part processes the query result data and internally generates the final search filtering options. The following is an example of a filtering option definition generated by the query: *+classname;DancingGoat.Coffee;Coffee*


This option appears as a "Coffee" checkbox in the filter. When selected, the filter ensures that the search results only display pages whose value in the *classname* field matches *DancingGoat.Coffee* (i.e. product pages of the *Coffee* page type).

Loading filtering options with non-string values

The second product filter provides a drop-down selector containing all available manufacturers. Add another **Smart search filter** web part to your search results page with the following properties:

- **Filter mode:** Dropdown list
- **Query name:** Create a **New** query with the following code:

```
SELECT ' ', ' ', '(Select Manufacturer)' as ManufacturerDisplayName
UNION
SELECT 'SKUManufacturerID', '(int)' + CAST(ManufacturerID AS VARCHAR(10)),
ManufacturerDisplayName
FROM COM_Manufacturer
WHERE ##WHERE##
ORDER BY ##ORDERBY##
```

 The first SELECT statement in the query prepares a static (*Select Manufacturer*) option without any filtering restrictions. The static option is then combined with the dynamically loaded options using a UNION operator.

Like with static filtering options, the data type of values must be specified for **integer**, **floating-point(double)** or **decimal** type fields (such as *SKUManufacturerID*). The *'(int)' + CAST(ManufacturerID AS VARCHAR(10))* part of the query loads the manufacturer ID values and adds the *(int)* prefix. When using SQL Server 2012 or newer, you can call the [CONCAT](#) SQL method instead, for example: *CONCAT('(int)', ManufacturerID)*

- **Query WHERE condition:** ManufacturerEnabled = 1
- **Query ORDER BY clause:** ManufacturerDisplayName
- **Filter clause:** Must
- **Filter is conditional:** Yes (selected)

The query fills the drop-down filter with a list of all manufacturers in the database and a static (*Select Manufacturer*) option. The filter ensures that the search results only display product pages whose manufacturer matches the selected option.

Filtering without search text (Faceted search)

You can implement a faceted search based exclusively on filters. Faceted search allows users to get search results simply by selecting filtering options, without the need to enter and submit search text. To achieve this result, **Configure** the following properties for your [search web parts](#):

Smart search results or Smart search dialog with results

- **Block fieldonly search:** Disabled (unless you wish to force users to use a filtering option that works with standard content fields, i.e. has *_content* as the field name)
- **Search on each page load:** Enabled (ensures that the web part automatically displays results whenever the page is loaded, even without input from a search dialog)
- **Search text required:** Disabled

Smart search filter

- **Filter auto postback:** Enabled (instantly refreshes the search results after users change the filtering options)