# Advanced Searching

Last Modified on 02/01/2024 9:49 am CST

## Purpose

The purpose of advanced searching is to allow users to construct their own detailed and complex search queries. Advanced searching allows grouping of criteria as well as exclusionary factors as well as many different types of comparisons based on the type of field and how it is stored in the database.

This article covers the following:



---

## Structure of an Advanced Search Request

Below is an example of an advanced search request for employees that searches for all employees that are assigned and that do not have an I-9 or have an expired I-9 before September 29th 2018. Requests are made to the search/{originTypeId} endpoint.

```
{
   "searchCriteria": {
      "rootGroup": {
         "condition": 0,
         "not": false,
         "rules": [{
            "columnId": "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568",
            "operatorId": 20,
            "values": null
         },
         {
            "condition": 1,
            "not": false,
            "rules": [{
               "condition": 0,
               "not": true,
               "rules": [{
                  "columnId": "19767911-09af-4998-bb51-b116204a25a4",
                  "operatorId": 7,
                  "values": ["15"]
               }]
            },
            {
               "condition": 0,
               "not": false,
               "rules": [{
                  "columnId": "19767911-09af-4998-bb51-b116204a25a4",
                  "operatorId": 7,
                  "values": ["15"]
               },
               {
                  "columnId": "654c9194-127b-453d-b49e-6170e5b4e2cc",
                  "operatorId": 4,
                  "values": ["2018-09-25T00:00:00"]
               }]
            }]
         }]
      },
      "returnColumnIds": ["1c587fca-b5b0-48dc-84a7-182469c0ee4f",
      "d14bf8a7-ba46-401d-8b7c-a264b117030a",
      "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
      "4f761764-c3ed-408f-8187-49fbf444df37",
      "9936c216-673b-47ee-b653-c1b2557183db",
      "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
      "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568",
      "7b434cb2-2c8b-4aa9-9c43-590d14185748",
      "20306c69-fd96-4e3a-a598-f36d38ef5f0b",
      "e4db9dfa-6f08-47a6-9dbe-555d1de5db5a"]
   },
   "sortByColumns": [{
      "columnId": "d14bf8a7-ba46-401d-8b7c-a264b117030a",
      "isDescending": false
   },
   {
      "columnId": "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
      "isDescending": false
   }],
   "ignoreCachedResults": false,
   "skip": 0,
   "take": 10
}
```


The request is broken up into four main parts, the search criteria, desired sort order, paging information, and

caching information.

## Search Criteria

The searchCriteria object contains two objects. The first is the list of columns to be included in the results and the second is the filtering to be performed to achieve the desired results.

### Return Columns

returnColumnIds is an array of guids (Globally/Universally Unique Identifiers). This list can be populated with values from the search/{originTypeId}/columns endpoint that are marked as canBeIncludedInResults. The order columns appear in this list is the order in which they will be returned in the results.

### Filter Criteria

rootGroup is the top level group where search criteria are specified. There are two different types of search criteria. Rules and Group.

### Groups

Groups are collections of other groups and rules. They consist of 3 parts.

```
{
   "condition": 0,
   "not": false,
   "rules": []
}
```

1. condition is used to specify whether or not all criteria in them must be met, 0, or at least one must be met,1. You can think of these as AND and OR logical operators applied between the groups and rules listed in rules.
2. rules is an array of rules and groups. If a group is one of the entries in this array, then the results of the group will be calculated before any other rules in the group. The system will then use the appropriate logic to apply the condition between the results of the records that matched that groups rules and the rest of the rules in the group.
3. not is used to specify whether or not the group is an exclusionary group. If set to true, then the set of valid results will be all results that do not match the rules and groups contained in rules. Please keep in mind that logically not not makes a true so be careful if you nest exclusionary groups underneath each other.

### Rules

Rules are filters on an individual column. They consist of three parts.

```
{
   "columnId": "19767911-09af-4998-bb51-b116204a25a4",
   "operatorId": 7,
   "values": ["15"]
}
```

1. columnId is the column to be filtered. A list of available columns can be found at search/{originTypeId}/columns.

2. operatorId is what logical filtering will be applied to the column. To determine which operators are valid for a given type of column, reference the search/{originTypeId}/columns and search/columnTypes endpoints. The operator will include information as to how many times it may be used in the current group and whether or not the column may be filtered again in the group. These values are dependent on the condition of the group the rule belongs to.
3. values is an array of values used for filtering. Most rules will have only zero or one item in this array. To know how many items to use, reference the expectedNumberOfValues field on the operator being used. If a column has a datalistId then a list of valid values for the column may be found at search/datalists/{datalistId}. A list of operators can be found at search/operators or search/columnTypes. Most operators take zero or one value. Those that take more are the following:
    - Between and Not Between each take two and the first value is the lower bound and the second is the upper bound. This means that a values array for Between of ["2017-07-01","2017-12-31"] for the column DateCreated will translates to DateCreated BETWEEN 2017-07-01 AND 2017-12-31 and will match all records created on or after 2017-07-01 but not after 2017-12-31. Please note that these two operators are inclusive. So in the given example, records created on 2017-07-01 or 2017-12-31 will be valid to be included in the results.
    - Distance from Postal Code takes three values. The first is the Postal Code, the second is the distance from the postal code, and the thrid is the unit of distance. Valid units of distance are mi for miles and km for kilometers. The system will default to kilometers if it cannot determine the units from the value provided. This means that a values array of ["55122", 10, "mi"] for the column PostalCode translates to PostalCode WITHIN 10 MILES OF 55122 and will include all records whose postal code within 10 miles of the postal code 55122.
    - Matches Any and Matches None require at least one value and can support variable numbers of values. So a values array for Matches None of [1,2,3,4,5] for the column EmployeeId will translate to EmployeeId NOT IN (1,2,3,4,5) and will match any record with one of the provided EmployeeIds.

### Sorting

SortByColumns is an array of columns and their sort directions that will be applied to the final results of the search before they are passed back. Each item in the sort array contains two fields.

```
{
  "columnId": "d14bf8a7-ba46-401d-8b7c-a264b117030a",
  "isDescending": false
}
```

1. columnId is the column that will be sorted on. The column must also be in the returned columns list. A list of columns can be obtained from the search/{originTypeId}/columns endpoint.
2. isDescending is the direction of the sort for the column. Setting this to true will make it so that larger values appear first.

### Paging

Search results are intended to be paged. As such, we require two fields to be provided.

1. skip is the number of results to skip over. Start at 0 to get the first page of results. Increase this by the page size to get each new successive page. So if your page size is 20, you will want to increase skip by 20 to get each new page.
2. take is your page size. This is the number of results that will be returned in each response. If this is 20, you will get 20 results back.

### Caching

Search results are cached if the number of results is less than the value specified in the database configuration fxMaxSearchRows. In most cases this defaults to 1000. The ability to cache results speeds up repeated requests with the same criteria where only the sorting or paging information changes. If you wish to avoid previously cached results and want fresh results, set ignoreCachedResults to true. Please note that if set this to true the search will be rerun which will decrease performance if you are simply paging through results.

---

## Structure of Advanced Search Results

Below is an example of an advanced search result for employees. Only one result has been included for the sake of brevity.


```
{
  "columns": [{
    "columnId": "1c587fca-b5b0-48dc-84a7-182469c0ee4f",
    "displayName": "ID",
    "columnType": "integer"
  },
  {
    "columnId": "d14bf8a7-ba46-401d-8b7c-a264b117030a",
    "displayName": "Last Name",
    "columnType": "string"
  },
  {
    "columnId": "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
    "displayName": "First Name",
    "columnType": "string"
  },
  {
    "columnId": "4f761764-c3ed-408f-8187-49fbf444df37",
    "displayName": "Branch",
    "columnType": "string"
  },
  {
    "columnId": "9936c216-673b-47ee-b653-c1b2557183db",
    "displayName": "Phone",
    "columnType": "string"
  },
  {
    "columnId": "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
    "displayName": "Is Active",
    "columnType": "boolean"
  },
  {
```

```
    "columnId": "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568",
    "displayName": "Is Assigned",
    "columnType": "boolean"
  },
  {
    "columnId": "7b434cb2-2c8b-4aa9-9c43-590d14185748",
    "displayName": "Last Message",
    "columnType": "string"
  },
  {
    "columnId": "20306c69-fd96-4e3a-a598-f36d38ef5f0b",
    "displayName": "Zip Code",
    "columnType": "postalcode"
  },
  {
    "columnId": "e4db9dfa-6f08-47a6-9dbe-555d1de5db5a",
    "displayName": "HasResume",
    "columnType": "boolean"
  }],
  "summary": "Document Type, Expiration Date, Is Assigned",
  "data": [{
    "Id": {
      "value": 4295064424,
      "originId": 4295064424,
      "originTypeId": 1
    },
    "1c587fca-b5b0-48dc-84a7-182469c0ee4f": {
      "value": 4295064424,
      "originId": 4295064424,
      "originTypeId": 1
    },
    "d14bf8a7-ba46-401d-8b7c-a264b117030a": {
      "value": "1",
      "originId": 4295064424,
      "originTypeId": 1
    },
    "2a9ab5d8-a9d8-49c8-ae1c-17737471f860": {
      "value": "Paytests",
      "originId": 4295064424,
      "originTypeId": 1
    },
    "4f761764-c3ed-408f-8187-49fbf444df37": {
      "value": "Memphis SE",
      "originId": null,
      "originTypeId": null
    },
    "9936c216-673b-47ee-b653-c1b2557183db": {
      "value": "1111111111",
      "originId": null,
      "originTypeId": null
    },
    "f7daae6e-91c2-411e-9d94-2c6f17725eb4": {
      "value": true,
      "originId": null,
      "originTypeId": null
    },
    "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568": {
      "value": true,
      "originId": null,
      "originTypeId": null
    },
    "7b434cb2-2c8b-4aa9-9c43-590d14185748": {
      "value": "Offered",
      "originId": null,
```

```
        "originTypeId": null
      },
      "20306c69-fd96-4e3a-a598-f36d38ef5f0b": {
        "value": "55122",
        "originId": null,
        "originTypeId": null
      },
      "e4db9dfa-6f08-47a6-9dbe-555d1de5db5a": {
        "value": true,
        "originId": null,
        "originTypeId": null
      }
    }],
    "totalCount": 1514
}
```


### Columns

columns is the list of returned columns. Each column contains three fields.


```
{
    "columnId": "7b434cb2-2c8b-4aa9-9c43-590d14185748",
    "displayName": "Last Message",
    "columnType": "string"
}
```


1. columnId is the Id of the column.
2. displayName is the human formatted label for the column as configured in the TempWorks system.
3. columnType is the data type of the data stored in the column.

### Results

data is a list that contains a bunch of dictionary objects that represent each result. Each result entry will contain at least a key of Id that contains the primary identifier (EmployeeId, CustomerId, etc.) for each matching result. The rest of the fields in the dictionary will have a key equal to that of their matching columnId. Each field in the result contains 3 fields.


```
{
    "Id": {
        "value": 4295064424,
        "originId": 4295064424,
        "originTypeId": 1
    },
    "1c587fca-b5b0-48dc-84a7-182469c0ee4f": {
        "value": 4295064424,
        "originId": 4295064424,
        "originTypeId": 1
    }
}
```


1. value is the value of the field.
2. originTypeId is the type of record the field belongs to is referencing. So if the field brought back is the CustomerId on a Job Order search, this value would be 2 as CustomerId belongs to the Customer. (See Origin Types for more).

3. originId is the primary identifier of the referenced record if one exists.

originTypeId and originId are used primarily for creating links to associated records in search results.

### Additional Information on the Results

There are a few additional fields included on the response.

1. totalCount is the total number of results that matched the search criteria.
2. summary is a quick summarization of the columns used to filter the results.

---

## Endpoints

**POST search/{originTypeId}**

This is the primary searching endpoint. The Origin Type determines which type of record is being searched. This endpoint takes in the advanced search criteria as the request body and returns the advanced search results as the results in the response body.

**GET search/{originTypeId}/columns**

This endpoint returns the columns available to filter on and return for the specified Origin Type. The following is an example of the results with descriptions of each field below:


```
{
   "data": [{
      "columnId": "fd7b554b-194e-4319-8f94-9cf7f9058a55",
      "categoryId": "65287139-9c2a-40f8-9bc2-2b490f838a05",
      "category": "ACA",
      "tableId": "d74d1f14-54d2-4481-adce-81199460e281",
      "columnType": "guid",
      "columnName": "EmploymentStatusId",
      "columnDisplayName": "ACA Status",
      "returnedColumnName": "EmploymentStatus",
      "returnedColumnType": "string",
      "datalistId": "b3e7bc1d-772d-4e7f-9883-74d2f61cffe3",
      "isNullable": true,
      "isCustomData": false,
      "isCustomDataJsonArray": false,
      "canBeIncludedInResults": true,
      "originTypeId": null,
      "originIdColumn": null
   }],
   "totalCount": 144
}
```


- columnId is the unique identifier for the column.
- categoryId is the category used to help users filter through results. This is not used when performing the search request.
- cateogry is a user readable display name for the category.

- tableId is a reference to the primary identifier on dbo.SearchTables. The table referenced is where the column exists in the database.
- columnType is the data type of the column that will be filtered on. If returnedColumnName is null then it is also the data type of the data that will be returned.
- columnName is the column on the table that will be used for filtering. If returnedColumnName is null then this is also the column that will be returned in results.
- columnDisplayName is the name that will be returned in the results as the user readable label for the column.
- returnedColumnName is an override for the column in the database that will be returned in the results. This is primarily used when filtering on an foreign key field where the value is not easy for users to understand. In the example column shown above, the columnType is a guid which users are not likely to recognize so we instead return its user readable value in the results.
- returnedColumnType is the data type of the column referenced in returnedColumnName. This value will be null if no value is in returnedColumnName
- datalistId is the unique identifier for the datalist of values that can be used to filter on the column. This should be used with the endpoint search/datalists/{datalistId} to get a list of valid values for the column.
- isNullable indicates whether or not the column can be nullable. If this value is false then you will not be able to use the Is Null and Not Null operators on the column.
- isCustomData indicates whether or not the column is pulled from our custom data system.
- isCustomDataJsonArray indicates whether or not the results for a custom data column are stored as a json array.
- originTypeId indicates whether or not the results will include reference information for a record. See the Results section for more information.
- originIdColumn is the column that will be used to populate originId field in the results of the column.

**GET search/datalists/{datalistId}**

This is the endpoint used to get valid values for a column that references a datalist. The following is an example of the results with descriptions of each field below:

```
{
  "data": [{
    "key": "566",
    "value": "11 Gold Express"
  },
  {
    "key": "671",
    "value": "12 hour Nurse"
  },
  {
    "key": "673",
    "value": "12 hour nurse weekend"
  },
  {
    "key": "346",
    "value": "2 Day Benefit"
  }],
  "totalCount": 720
}
```


- key is the field that will be supplied to the values array for a rule.
- value is a user readable string.

#### GET search/operators

This is the endpoint used to get a list of possible operators. The following is an example of the results with descriptions of each field below:


```
{
  "data": [{
    "operatorId": 1,
    "operator": "Less Than",
    "expectedNumberOfValues": "1",
    "isComplexOperator": false,
    "isUniquePerColumnInAndGroup": true,
    "makesColumnUniqueInAndGroup": false,
    "isUniquePerColumnInOrGroup": true,
    "makesColumnUniqueInOrGroup": false
  }],
  "totalCount": 19
}
```


- operatorId is the unique identifier for the operator and is the value that will be supplied to the operatorId field for a rule.
- operator is a user friendly description for the operator.
- expectedNumberOfValues is the number of values expected to be in the values aray for a rule. Valid values for this field are: 0, 1, 2, and many.
- isComplexOperator is a system flag used to indicate whether or not complex logic is performed to filter using this operator. It should not change anything about making a request.
- isUniquePerColumnInAndGroup indicates whether or not a column can be filtered on using this operator multiple times in a group with the condition set to 0. This is set to prevent users from constructing searches like the following EmployeeId EQUALS 1 AND EmployeeId EQUALS 2 where it is not possible to actually have any record match the combined criteria.

- makesColumnUniqueInAndGroup indicates whether or not a column can be filtered on multiple times in a group with the condition set to 0 if this operator is used. This is set to prevent users from constructing searches like the following FirstName EQUALS "Joe" AND FirstName STARTS WITH "jim%" where it is not possible to actually have any record match the combined criteria or makes further filtering redundant.
- isUniquePerColumnInOrGroup indicates whether or not a column can be filtered on using this operator multiple times in a group with the condition set to 1. This is set to prevent users from constructing searches like the following EmployeeId DOES NOT EQUAL 1 OR EmployeeId DOES NOT EQUAL 2 where all records automatically match the combined criteria.
- makesColumnUniqueInOrGroup indicates whether or not a column can be filtered on multiple times in a group with the condition set to 1 if this operator is used. This is set to prevent users from constructing searches like the following FirstName IS NOT NULL OR FirstName STARTS WITH "jim%" where the use of the operator makes all further filtering done on the column redundant.

**GET search/columnTypes**

This is the endpoint used to get a list of possible column types and their applicable operators. The following is an example of the results:


```
{
   "data": [{
      "type": "boolean",
      "operators": [{
         "operatorId": 14,
         "operator": "Is Null",
         "expectedNumberOfValues": "0",
         "isComplexOperator": false,
         "isUniquePerColumnInAndGroup": true,
         "makesColumnUniqueInAndGroup": true,
         "isUniquePerColumnInOrGroup": true,
         "makesColumnUniqueInOrGroup": false
      },
      {
         "operatorId": 15,
         "operator": "Not Null",
         "expectedNumberOfValues": "0",
         "isComplexOperator": false,
         "isUniquePerColumnInAndGroup": true,
         "makesColumnUniqueInAndGroup": false,
         "isUniquePerColumnInOrGroup": true,
         "makesColumnUniqueInOrGroup": true
      }]
   }],
   "totalCount": 11
}
```


*Note* Is Null and Not Null should only be presented to the user if isNullable is set to true on the column in the search/{originTypeId}/columns endpoint. For a description of the fields on each of the operators, please reference the search/operators endpoint.

## Origin Types

Many endpoints and objects contain a field or parameter named originTypeId. Origin Type is an Enum used to indicate the type of record referenced. Valid values for originTypeId and their corresponding record type are the following:

- 1: Employee
- 2: Customer
- 3: Contact
- 8: Job Order
- 14: Assignment

## How To Step-By-Step

*Note* This step-by-step assumes the columns you want to use have all been configured for advanced search. If additional columns are needed these should be requested from Tempworks.

### Step 1: Select Your Origin Type

This will just be placed into the URL and we will start with the request body below as our blank template.

```
https://api.ontempworks.com/Search/:originTypeId

{
  "searchCriteria": {
    "rootGroup": {
      "condition": 0,
      "not": false,
      "rules": []
    },
    "returnColumnIds": []
  },
  "sortByColumns": [],
  "ignoreCachedResults": ,
  "skip": ,
  "take":
}
```

### Step 2: Conceptualize Your Search

For this example, we will be finding active employees in branch 'High Tech SE' whoeither are flagged as having their I-9 on file, have an I-9 expiration date set, or have an attached document of type 'Federal I-9' with a document expiration after today.

### Step 3: Start Crafting the Query Body by Identify the Columns You Want to Include in Your Search "Rules" Inside Your "searchCriteria"

> *Note* See ### GET search/{originTypeId}/columns for finding the available columnIds.

On this endpoint you'll notice some are marked as 'canBeIncludedInResults' = false. This has to do with the table relationships being one-to-one or one-to-many with the origin type's table. This is important for step 6 when we are deciding which column to return.

- "f7daae6e-91c2-411e-9d94-2c6f17725eb4" = 'Is Active' flag on the employee
- "4f761764-c3ed-408f-8187-49fbf444df37" = 'Branch' the branch of the employee
- "8482430e-1a09-4030-97a1-77c6ab7ed299" = 'I-9 is On File'
- "e9e052e5-0c66-49d3-9a3c-21bdb1913004" = 'I-9 Expires'
- "19767911-09af-4998-bb51-b116204a25a4" = 'Document Type'
- "654c9194-127b-453d-b49e-6170e5b4e2cc" = 'Document Expiration Date'

### Step 4: Create Rule Groupings for Your Columns that Match the Your Search Concept

In the root group's rules, we want things that are One-to-one with the employee record and are all true. Since these things are all true, we want to use condition = 0 for an 'AND' operation.

- Is Active = true
- Branch = 'High Tech SE'
- A sub-grouping for our I-9 rules: In this group we are looking for any of the following rules to be true. So we would use condition = 1 for an 'OR' operation.
    - 'I-9 is On File'
    - 'I-9 Expires'
    - A sub-grouping for the document search. Because documents are One-to-Many with employees, we need a sub-grouping here to only include documents that match the document type we are looking for. We use condition = 0 here because we want docuemnts that are of type 'Federal I-9' AND expire after todays date.
        - 'Document Type'
        - 'Document Expiration Date'

```
{
 "searchCriteria": {
  "rootGroup": {
   "condition": 0,
   "not": false,
   "rules": [
    {
     "columnId": "f7daae6e-91c2-411e-9d94-2c6f17725eb4"
    },
    {
     "columnId": "4f761764-c3ed-408f-8187-49fbf444df37"
    },
    {
     "condition": 1,
     "not": false,
     "rules": [
      {
       "columnId": "8482430e-1a09-4030-97a1-77c6ab7ed299"
      },
      {
       "columnId": "e9e052e5-0c66-49d3-9a3c-21bdb1913004"
      },
      {
       "condition": 0,
       "not": false,
       "rules": [
        {
         "columnId": "19767911-09af-4998-bb51-b116204a25a4"
        },
        {
         "columnId": "654c9194-127b-453d-b49e-6170e5b4e2cc"
        }
       ]
      }
     ]
    }
   ]
  },
  "returnColumnIds": []
 },
 "sortByColumns": [],
 "ignoreCachedResults": ,
 "skip": ,
 "take":
}
```

Step 5: Use the 'columnType' of your selected rule columns and the ### GET search/columnTypes endpoint to find an appropriate operatorId then fill the 'values' property approprately.

```
{
 "searchCriteria": {
  "rootGroup": {
   "condition": 0,
   "not": false,
   "rules": [
    {
     "columnId": "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
     "operatorId": 20,
     "values": null
    },
    {
     "columnId": "4f761764-c3ed-408f-8187-49fbf444df37",
     "operatorId": 7,
```

```
        "values": [
          "High Tech SE"
        ]
       },
       {
        "condition": 1,
        "not": false,
        "rules": [
         {
          "columnId": "8482430e-1a09-4030-97a1-77c6ab7ed299",
          "operatorId": 20,
          "values": null
         },
         {
          "columnId": "e9e052e5-0c66-49d3-9a3c-21bdb1913004",
          "operatorId": 2,
          "values": [
           "2022-01-04T00:00:00-06:00"
          ]
         },
         {
          "condition": 0,
          "not": false,
          "rules": [
           {
            "columnId": "19767911-09af-4998-bb51-b116204a25a4",
            "operatorId": 7,
            "values": [
             "15"
            ]
           },
           {
            "columnId": "654c9194-127b-453d-b49e-6170e5b4e2cc",
            "operatorId": 2,
            "values": [
             "2024-01-05T00:00:00-06:00"
            ]
           }
          ]
         }
        ]
       }
      ]
     },
     "returnColumnIds": []
    },
    "sortByColumns": [],
    "ignoreCachedResults": ,
    "skip": ,
    "take":
}
```


Step 6: Identify the columns you want to be retreived from your search and add their guids to "returnColumnIds" inside the "searchCriteria".

We'll just be returning some standard employee informaiton columns.  This completes our "searchCriteria":

- "1c587fca-b5b0-48dc-84a7-182469c0ee4f"
- "d14bf8a7-ba46-401d-8b7c-a264b117030a"
- "2a9ab5d8-a9d8-49c8-ae1c-17737471f860"

- "4f761764-c3ed-408f-8187-49fbf444df37"
- "9936c216-673b-47ee-b653-c1b2557183db"
- "647e7c49-2546-4a30-9ea2-916c5beca596"
- "0683a916-7ed3-4961-9300-474bd586f589"
- "f7daae6e-91c2-411e-9d94-2c6f17725eb4"
- "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568"
- "7b434cb2-2c8b-4aa9-9c43-590d14185748"
- "20306c69-fd96-4e3a-a598-f36d38ef5f0b"
- "e4db9dfa-6f08-47a6-9dbe-555d1de5db5a"
- "8482430e-1a09-4030-97a1-77c6ab7ed299"
- "e9e052e5-0c66-49d3-9a3c-21bdb1913004"


```
{
 "searchCriteria": {
  "rootGroup": {
   "condition": 0,
   "not": false,
   "rules": [
    {
     "columnId": "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
     "operatorId": 20,
     "values": null
    },
    {
     "columnId": "4f761764-c3ed-408f-8187-49fbf444df37",
     "operatorId": 7,
     "values": [
      "High Tech SE"
     ]
    },
    {
     "condition": 1,
     "not": false,
     "rules": [
      {
       "columnId": "8482430e-1a09-4030-97a1-77c6ab7ed299",
       "operatorId": 20,
       "values": null
      },
      {
       "columnId": "e9e052e5-0c66-49d3-9a3c-21bdb1913004",
       "operatorId": 2,
       "values": [
        "2022-01-04T00:00:00-06:00"
       ]
      },
      {
       "condition": 0,
       "not": false,
       "rules": [
        {
         "columnId": "19767911-09af-4998-bb51-b116204a25a4",
         "operatorId": 7,
         "values": [
          "15"
         ]
        },
```

```
          {
            "columnId": "654c9194-127b-453d-b49e-6170e5b4e2cc",
            "operatorId": 2,
            "values": [
              "2024-01-05T00:00:00-06:00"
            ]
          }
        ]
      }
    ]
   }
  ],
  "returnColumnIds": [
    "1c587fca-b5b0-48dc-84a7-182469c0ee4f",
    "d14bf8a7-ba46-401d-8b7c-a264b117030a",
    "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
    "4f761764-c3ed-408f-8187-49fbf444df37",
    "9936c216-673b-47ee-b653-c1b2557183db",
    "647e7c49-2546-4a30-9ea2-916c5beca596",
    "0683a916-7ed3-4961-9300-474bd586f589",
    "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
    "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568",
    "7b434cb2-2c8b-4aa9-9c43-590d14185748",
    "20306c69-fd96-4e3a-a598-f36d38ef5f0b",
    "e4db9dfa-6f08-47a6-9dbe-555d1de5db5a",
    "8482430e-1a09-4030-97a1-77c6ab7ed299",
    "e9e052e5-0c66-49d3-9a3c-21bdb1913004"
  ]
 },
 "sortByColumns": [],
 "ignoreCachedResults": ,
 "skip": ,
 "take":
}
```

### Step 7: Identify the order you want to use to sort your search results by and add their guids and sort order as "columnId" and "isDescending" to "sortByColumns"

```
"sortByColumns": [
  {
    "columnId": "d14bf8a7-ba46-401d-8b7c-a264b117030a",
    "isDescending": false
  },
  {
    "columnId": "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
    "isDescending": false
  }
]
```

### Step 8: Finally, set values for "ignoreCachedResults" (boolean), "skip" (int), and "take" (int) appropriately.

This completes our request body.

- ignoreCachedResults (boolean): Whether we are ignoring previously cached results (see ### Caching). For performance reasons, this should only be true when re-doing a recently run search when you want to ignore the previous results. For new searches and for paging it should be set to false.
- skip (int): The nujber of results you need to skip to reach the page you are trying to access.

- take (int): The number of results you want to include in this page of search results.


```
{
 "searchCriteria": {
  "rootGroup": {
   "condition": 0,
   "not": false,
   "rules": [
    {
     "columnId": "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
     "operatorId": 20,
     "values": null
    },
    {
     "columnId": "4f761764-c3ed-408f-8187-49fbf444df37",
     "operatorId": 7,
     "values": [
      "High Tech SE"
     ]
    },
    {
     "condition": 1,
     "not": false,
     "rules": [
      {
       "columnId": "8482430e-1a09-4030-97a1-77c6ab7ed299",
       "operatorId": 20,
       "values": null
      },
      {
       "columnId": "e9e052e5-0c66-49d3-9a3c-21bdb1913004",
       "operatorId": 2,
       "values": [
        "2022-01-04T00:00:00-06:00"
       ]
      },
      {
       "condition": 0,
       "not": false,
       "rules": [
        {
         "columnId": "19767911-09af-4998-bb51-b116204a25a4",
         "operatorId": 7,
         "values": [
          "15"
         ]
        },
        {
         "columnId": "654c9194-127b-453d-b49e-6170e5b4e2cc",
         "operatorId": 2,
         "values": [
          "2024-01-05T00:00:00-06:00"
         ]
        }
       ]
      }
     ]
    }
   ]
  },
  "returnColumnIds": [
   "1c587fca-b5b0-48dc-84a7-182469c0ee4f",
   "d14bf8a7-ba46-401d-8b7c-a264b117030a",
```

```
      "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
      "4f761764-c3ed-408f-8187-49fbf444df37",
      "9936c216-673b-47ee-b653-c1b2557183db",
      "647e7c49-2546-4a30-9ea2-916c5beca596",
      "0683a916-7ed3-4961-9300-474bd586f589",
      "f7daae6e-91c2-411e-9d94-2c6f17725eb4",
      "8d2aa3b9-2aaf-43c7-9646-3cf910a3a568",
      "7b434cb2-2c8b-4aa9-9c43-590d14185748",
      "20306c69-fd96-4e3a-a598-f36d38ef5f0b",
      "e4db9dfa-6f08-47a6-9dbe-555d1de5db5a",
      "8482430e-1a09-4030-97a1-77c6ab7ed299",
      "e9e052e5-0c66-49d3-9a3c-21bdb1913004"
    ]
  },
  "sortByColumns": [
    {
      "columnId": "d14bf8a7-ba46-401d-8b7c-a264b117030a",
      "isDescending": false
    },
    {
      "columnId": "2a9ab5d8-a9d8-49c8-ae1c-17737471f860",
      "isDescending": false
    }
  ],
  "ignoreCachedResults": false,
  "skip": 0,
  "take": 100
}
```