Supported Script Types	Client scripts For more information, see SuiteScript 2.0 Client Script Type.
Governance	None
Module	N/portlet Module
Since	2016.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/portlet Module Script Sample.

```
//Add additional code
...
portlet.resize();
...
//Add additional code
```

portlet.refresh

Method Description	Refreshes a form portlet type immediately.
Returns	Void
Supported Script Types	Client scripts For more information, see SuiteScript 2.0 Client Script Type.
Governance	None
Module	N/portlet Module
Since	2016.1

Syntax

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/portlet Module Script Sample.

```
...
portlet.refresh();
...
```

N/query Module

Load the query module to create and run queries using the SuiteAnalytics Workbook query engine. For more information, see the help topic SuiteAnalytics Workbook Beta. Using the query module, you can:

- Use multilevel joins to create queries using field data from multiple record types.
- Create conditions (filters) using AND, OR, and NOT logic, as well as formulas.
- Sort query results based on the values of multiple columns.
- Load and delete existing saved queries that were created using the SuiteAnalytics Workbook UI.
- View paged query results.
- Use promises for asynchronous execution.

Important: The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. In the 2018.2 release, you can use the N/query module to load and delete existing searches, but you cannot save searches. You can save searches using the SuiteAnalytics Workbook UI.

For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For information, see the help topic Supported Record Types for the SuiteAnalytics Workbook Beta Period.

N/query Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	query.Column	Object	Client and server-side scripts	Encapsulates the field types (query result columns) that are displayed from the query results. Use Query.createColumn(options) or Component.createColumn(options) to create this object.
	query.Component	Object	Client and server-side scripts	Encapsulates one component of the query definition. The query definition always contains at least one component that encapsulates the initial search type. Queries with joins contain multiple components that encapsulate the join relationships. The initial component (Query.root) is automatically created with the query definition (query.Query). Use Query.autoJoin(options) or Component.autoJoin(options) to create subsequent components.
	query.Condition	Object	Client and server-side scripts	Encapsulates a condition. A condition narrows the query results. Use Query.createCondition(options) or Component.createCondition(options) to create this object.
	query.Page	Object	Client and server-side scripts	Encapsulates one page of the paged query results.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	query.PagedData	Object	Client and server-side scripts	Encapsulates a set of paged query results. This object also contains information about the set of paged results it encapsulates.
	query.PageRange	Object	Client and server-side scripts	Encapsulates a range of pages from the paged query results.
	query.Result	Object	Client and server-side scripts	Encapsulates a single row of the query result set.
	query.ResultSet	Object	Client and server-side scripts	Encapsulates the set of results returned by the query.
	query.Query	Object	Client and server-side scripts	Encapsulates the query definition. Use query.create(options) or query.load(options) to create this object. **The creation of this object is the first step in creating a query with the N/query Module**.
	query.Sort	Object	Client and server-side scripts	Encapsulates a sort that is placed on a particular query result column. Use Query.createSort(options) or Component.createSort(options) to create this object.
Method	query.create(options)	query.Query	Client and server-side scripts	Creates the query definition. **The execution of this method is the first step in creating a query with the N/query Module**.
	query.delete(options)	void	Client and server-side scripts	Deletes an existing query that was created using the SuiteAnalytics Workbook UI. The deleted query is no longer available and cannot be modified or executed.
	query.load(options)	query.Query	Client and server-side scripts	Loads an existing query that was created using the SuiteAnalytics Workbook UI. The loaded query can be modified (for example, by setting additional property values), joined with other search types, and executed in the same way as queries created using query.create(options).
Enum	query.Aggregate	enum	Client and server-side scripts	Holds the string values for aggregate functions supported with the N/query Module. This enum is used to pass the aggregate function argument to Component.createColumn(options), Component.createCondition(options), Query.createColumn(options), and Query.createCondition(options).
	query.Operator	enum	Client and server-side scripts	Holds the string values for operators supported with the N/query Module. This enum is used to pass the operator argument to

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				Query.createCondition(options) and Component.createCondition(options).
	query.ReturnType	enum	Client and server-side scripts	Holds the string values for the formula return types supported with the N/query Module. This enum is used to pass the formula return type argument to Query.createColumn(options), Component.createColumn(options), Query.createCondition(options), and Component.createCondition(options).
	query.SortLocale	enum	Client and server-side scripts	Holds the string values for sort locales supported with the N/query Module. This enum is used to pass the sort locale argument to Query.createSort(options) and Component.createSort(options).
	query.Type	enum	Client and server-side scripts	Holds the string values for supported search types used in the query definition. This enum is used to pass the initial search type argument to query.create(options).

Column Object Members

The following members are called on the query.Column object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Column.aggregate	string (read-only)	Client and server-side scripts	Describes an aggregate function that is performed on the query result column. An aggregate function performs a calculation on the column values and returns a single value.
	Column.component	query.Component (read-only)	Client and server-side scripts	Holds a reference to the query.Component object to which this query result column belongs.
	Column.fieldId	string (read-only)	Client and server-side scripts	Holds the name of the query result column. This property and the Column.formula property cannot be set at the same time.
	Column.formula	string (read-only)	Client and server-side scripts	Describes the formula used to create the query result column. This property and the Column.fieldId property cannot be set at the same time.
	Column.groupBy	boolean (read-only)	Client and server-side scripts	Indicates whether the query results are grouped by this query result column.
	Column.type	string (read-only)	Client and server-side scripts	Describes the return type of the formula used to create the query result column.

Component Object Members

The following members are called on the query.Component object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	Component.autoJoin(options)	query.Component	Client and server-side scripts	Creates a join relationship. After you create the initial query definition, use Query.autoJoin(options) to create your first join. Then use this method to create each subsequent join.
	Component.createColumn(options)	query.Column	Client and server-side scripts	Creates a query result column based on the component. Use this method to create columns based on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options).
	Component.createCondition(options)	query.Condition	Client and server-side scripts	Creates a condition (filter column) based on the component. Use this method to create conditions based on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options).
	Component.createSort(options)	query.Sort	Client and server-side scripts	Creates a sort based on the component. Use this method to create sorts based on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options).
	Component.join(options)	query.Component	Client and server-side scripts	Creates a join relationship. This method is an alias to Component.autoJoin(options). After you create the initial query definition, use Query.autoJoin(options) to create your first join. Then use this method, or Component.autoJoin(options), to create each subsequent join.
	Component.joinFrom(options)	query.Component	Client and server-side scripts	Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the Component.source property on the returned query.Component object. After you create the initial query definition, use this method to create explicit directional joins from other components to this component.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
	Component.joinTo(options)	query.Component	Client and server-side scripts	Creates an explicit directional join relationship to another component from this component (a polymorphic join). You can use this method to specify the target of the join when a field can join multiple search types. This method sets the Component.target property on the returned query.Component object. After you create the initial query definition, use this method to create explicit directional joins to other components from this component.
Property	Component.child	Object (read-only)	Client and server-side scripts	Describes the child components of the component. This property holds an object of key/value pairs. Each key is the name of a child component. Each value is the corresponding child query.Component object.
	Component.parent	string (read-only)	Client and server-side scripts	Describes the parent query.Component object of the component.
	Component.source	string (read-only)	Client and server-side scripts	Describes the source search type of the component. The value of this property is set when Component.joinFrom(options) is called to perform an explicit directional join from another component.
	Component.target	string (read-only)	Client and server-side scripts	Describes the target search type of the component. The value of this property is set when Component.joinTo(options) is called to perform an explicit directional join to another component.
	Component.type	string (read-only)	Client and server-side scripts	Describes the search type of the component.

Condition Object Members

The following members are called on the query.Condition object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Condition.aggregate	string (read-only)	Client and server-side scripts	Describes an aggregate function that is performed on the condition. An aggregate function performs a calculation on the condition values and returns a single value.
	Condition.children	query.Condition[] (read-only)	Client and server-side scripts	Holds an array of child conditions used to create the parent condition.
	Condition.component	query.Component (read-only)	Client and server-side scripts	Holds a reference to the query.Component object to which this condition belongs.
	Condition.fieldId	string (read-only)	Client and server-side scripts	Holds the name of the condition.
	Condition.formula	string (read-only)	Client and server-side scripts	Describes the formula used to create the condition.
	Condition.operator	string (read-only)	Client and server-side scripts	Holds the name of the operator used to create the condition.
	Condition.type	string (read-only)	Client and server-side scripts	The return type of the formula used to create the condition.
	Condition.values	string[] (read-only)	Client and server-side scripts	Holds an array of values used by an operator to create the condition.

Page Object Members

The following members are called on the query.Page object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Page.data	query.ResultSet (read-only)	Client and server-side scripts	References the query results contained in this page.
	Page.isFirst	boolean (read-only)	Client and server-side scripts	Indicates whether this page is the first of the paged query results.
	Page.isLast	boolean (read-only)	Client and server-side scripts	Indicates whether this page is the last of the paged query results.
	Page.pagedData	query.PagedData (read-only)	Client and server-side scripts	References the set of paged query results that this page is from.
	Page.pageRange	query.PageRange (read-only)	Client and server-side scripts	The range of query results for this page.

PagedData Object Members

The following members are called on the query.PagedData object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	PagedData.iterator()	Iterator object	Client and server-side scripts	Standard SuiteScript 2.0 object for iterating through results.
Property	PagedData.count	number (read-only)	Client and server-side scripts	Describes the total number of paged query results.
	PagedData.pageRanges	query.PageRange[]	Client and server-side scripts	Holds an array of page ranges for the set of paged query results.
	PagedData.pageSize	number (read-only)	Client and server-side scripts	Describes the number of query result rows per page.

PageRange Object Members

The following members are called on the query.PageRange object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	PageRange.index	number (read-only)	Client and server-side scripts	Describes the array index for this page range.
	PageRange.size	number (read-only)	Client and server-side scripts	Describes the number of query result rows in this page range.

Query Object Members

The following members are called on the query.Query object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	Query.and()	query.Condition object	Client and server-side scripts	Creates a new condition (a query.Condition object) that corresponds to a logical conjunction (AND) of the arguments passed to the method. The arguments must be one or more query.Condition objects.
	Query.autoJoin(options)	query.Component	Client and server-side scripts	Creates a join relationship. After you create the initial query definition, use this method to create your first join.
	Query.createColumn(options)	query.Column object	Client and server-side scripts	Creates a query result column based on the query.Query object. Use this method to create columns on the initial query

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
				definition created with query.create(options).
	Query.createCondition(options)	query.Condition object	Client and server-side scripts	Creates a condition (filter column) based on the query.Query object. Use this method to create conditions on the initial query definition created with query.create(options).
	Query.createSort(options)	query.Sort object	Client and server-side scripts	Creates a sort based on the query.Query object. The query.Sort object describes a sort that is placed on a particular query result column or condition.
	Query.join(options)	query.Component	Client and server-side scripts	Creates a join relationship. This method is an alias to Query.autoJoin(options). After you create the initial query definition, use this method, or Query.autoJoin(options), to create your first join.
	Query.joinFrom(options)	query.Component	Client and server-side scripts	Creates an explicit directional join relationship from another component to the root component of the search definition (an inverse join). This method sets the Component.source property on the returned query.Component object. After you create the initial query definition, use this method to create your first join as an explicit directional join from another component to this component.
	Query.joinTo(options)	query.Component	Client and server-side scripts	Creates an explicit directional join relationship to another component from this component (a polymorphic join). You can use this method to specify the target of the join when a field can join multiple search types. This method sets the Component.target property on the returned query.Component object. After you create the initial query definition, use this method to create your first join as an explicit directional join to another component from this component.
	Query.not()	query.Condition	Client and server-side scripts	Creates a new condition (a query.Condition object) that corresponds to a logical

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
				negation (NOT) of the argument passed to the method. The argument must be a query.Condition object.
	Query.or()	query.Condition	Client and server-side scripts	Creates a new condition (a query.Condition object) that corresponds to a logical disjunction (OR) of the arguments passed to the method. The arguments must be one or more query.Condition objects.
	Query.run()	query.ResultSet	Client and server-side scripts	Executes the query and returns the query result set.
	Query.run.promise()	query.ResultSet	Client scripts	Executes the query asynchronously and returns the query result set.
	Query.runPaged()	query.PagedData	Client and server-side scripts	Executes the query and returns a set of paged results.
	Query.runPaged.promise()	query.PagedData	Client scripts	Executes the query asynchronously and returns a set of paged results.
Property	Query.child	Object (read-only)	Client and server-side scripts	Holds a references to children of the root component of the query definition. The value of this property is an object of key/value pairs. Each key is the name of a child component. Each respective value is the corresponding query.Component object.
	Query.columns	query.Column[]	Client and server-side scripts	Holds an array of query result columns returned from the query. Before you execute the query, you must assign all created columns as array values to this property.
	Query.condition	query.Condition object	Client and server-side scripts	References the parent condition that narrows the query results. Before you execute the query, you must assign your simple or complex conditions to this property.
	Query.id	number (read-only)	Client and server-side scripts	Holds the ID of the query definition. This property has a value only for existing queries that are loaded using query.load(options). If you create a query using

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
				query.create(options) but do not save it, this property is null.
	Query.name	string (read-only)	Client and server-side scripts	Holds the name of the query definition. This property has a value only for existing queries that are loaded using query.load(options). If you create a query using query.create(options) but do not save it, this property is null.
	Query.root	query.Component (read-only)	Client and server-side scripts	References the root component of the query definition.
	Query.sort	query.Column[] (read-only)	Client and server-side scripts	Holds an array of query result columns used for sorting.
	Query.type	string (read-only)	Client and server-side scripts	Holds the search type of the initial query definition.

Result Object Members

The following members are called on the query.Result object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Result.columns	query.Column[] (read-only)	Client and server-side scripts	Holds an array of query result column references.
	Result.values	string[] or number[] or boolean[] (read-only)	Client and server-side scripts	Describes the result values.

ResultSet Object Members

The following members are called on the query.ResultSet object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	ResultSet.iterator()	Iterator object	Client and server-side scripts	Standard SuiteScript 2.0 object for iterating through results.
Property	ResultSet.columns	query.Column[] (read-only)	Client and server-side scripts	Holds an array of query result column references.
	ResultSet.results	query.Result[] (read-only)	Client and server-side scripts	Holds an array of query.Result objects.
	ResultSet.types	string[] (read-only)	Client and server-side scripts	Holds an array of the return types for ResultSet.results.

Sort Object Members

The following members are called on the query.Sort object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Sort.ascending	boolean	Client and server-side scripts	Indicates whether the sort direction is ascending.
	Sort.caseSensitive	boolean	Client and server-side scripts	Indicates whether the sort is case sensitive. If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same.
	Sort.column	query.Column (read-only)	Client and server-side scripts	Describes the query result column that the query results are sorted by.
	Sort.locale	string	Client and server-side scripts	The locale to use for the sort. A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted.
	Sort.nullsLast	boolean	Client and server-side scripts	Indicates whether query results with null values are listed at the end of the query results.

N/query Module Script Samples

```
require(['N/query'],
    function(query) {
        var search = query.create({
            type: query.Type.CUSTOMER
        });

         var salesrep = search.autoJoin({
            fieldId: 'salesrep'
        });
         var location = salesrep.autoJoin({
            fieldId: 'location'
        });

         var cond1 = search.createCondition({
            fieldId: 'id',
            operator: query.Operator.EQUAL,
            values: 107
        });
         var cond2 = search.createCondition({
            fieldId: 'id',
            operator: query.Operator.EQUAL,
            values: 2647
        });
         var cond3 = salesrep.createCondition({
            fieldId: 'email',
            operator: query.Operator.START_WITH_NOT,
            values: 'foo'
        });
         search.condition = search.and(
            cond3, search.or(cond1, cond2)
        );

         search.columns = [
              search.createColumn({
                fieldId: 'entityid'
            }),
              search.createColumn({
                fieldId: 'id'
            }),
              salesrep.createColumn({
                fieldId: 'entityid'
            }),
              salesrep.createColumn({
                fieldId: 'email'
            }),
              salesrep.createColumn({
                fieldId: 'hiredate'
            }),
              location.createColumn({
                fieldId: 'name'
            })
        ];
         search.sort = [
              search.createSort({
                column: search.columns[3]
            }),
              search.createSort({
                column: search.columns[0],
                ascending: false
            })
        ];

         var resultSet = search.run();

         var results = resultSet.results;
         for (var i = results.length - 1; i >== 0; i--)
              log.debug(results[i].values);
         log.debug(resultSet.types);

         log.error(
            search.root === location.parent.parent
        );
         log.error(
            search.root.child.salesrep === location.parent
        );
         log.error(
            search.child.salesrep === location.parent
```

Scripting with the N/query Module

The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. Before you start creating your queries, you should be familiar with the module objects and how to use them, as well as some of the terminology used in the N/query module. You can also take a look at a script walkthrough that explains how to create queries using different approaches.

- N/query Module Objects

- N/query Module Terminology
- N/query Module Script Walkthrough

N/query Module Objects

The N/query module includes the following objects:

- Query and Component Objects
- Condition Object
- Column Object
- Sort Object
- ResultSet and Result Objects
- Page, PagedData, and PageRange Objects

Query and Component Objects

The query.Query object and the query.Component object are the primary building blocks for a query created with the N/query module. Each query creates one query.Query object and one or more query.Component objects. The query.Query object encapsulates the query definition, and the query.Component object encapsulates one component of the query definition.

To create a query with the N/query module:

1. Use the query.create(options) method to create your initial query definition (the query.Query object). The initial query definition uses one search type. For available search types, see query.Type.
2. After you create the initial query definition, use Query.autoJoin(options), Query.joinFrom(options), or Query.joinTo(options) to create your first join.
3. Use Component.autoJoin(options), Component.joinFrom(options), or Component.joinTo(options) to create all subsequent joins.

The query definition always contains at least one query.Component object. Each new component is created as a child of the previous component, and all components exist as children of the query definition. You can think of a component as a building block; each new component builds on the previous component created. The last component created encapsulates the relationship between it and all of its parent components.

Queries with joins contain multiple components. The query definition contains a child query.Component object for each of the following:

- **The initial query definition:** The initial query.Component object is called the root component. It encapsulates the initial search type passed to query.create(options). The root component is automatically created with the initial query definition and is a child to the query.Query object. The Query.root property contains a reference to the root component.
- **The first join:** The second query.Component object is created with Query.autoJoin(options), Query.joinFrom(options), or Query.joinTo(options). It encapsulates the relationship between the initial query definition and the second search type. This relationship is determined by the join ID passed to these methods, as well as whether Query.joinFrom(options) or Query.joinTo(options) was used to create an explicit directional join. The second query.Component object is a child to the root component.
- **Each subsequent join:** The third query.Component object is created with Component.autoJoin(options), Component.joinFrom(options), or Component.joinTo(options). All subsequent joins are also created using these methods. Each of these query.Component objects encapsulates the relationship between all previous search types and the new search

type. This relationship is determined by the join ID passed to these methods, as well as whether Component.joinFrom(options) or Component.joinTo(options) was used to create an explicit directional join.

Condition Object

A condition narrows the query results. The query.Condition object performs the same function as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects.

To create conditions:

- Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options).
- Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options), Query.joinFrom(options)/Query.joinTo(options), Component.autoJoin(options), or Component.joinFrom(options)/Component.joinTo(options).
- If you have multiple conditions, use Query.and(), Query.or(), and Query.not() to create a new nested condition.
- If you want to use a formula to define your conditions, assign the formula to Condition.formula.
- Assign your simple or nested conditions as array values to Query.condition.

Column Object

The query.Column object is the equivalent of the search.Column object in the N/search Module. The query.Column object describes the field types (columns) that are displayed from the query results.

To create columns:

- Use Query.createColumn(options) to create a column on the initial query definition created with query.create(options).
- Use Component.createColumn(options) to create a column on a join relationship created with Query.autoJoin(options), Query.joinFrom(options)/Query.joinTo(options), Component.autoJoin(options), or Component.joinFrom(options)/Component.joinTo(options).
- If you want to use a formula to define your columns, assign the formula to Column.formula.
- Assign all created columns as array values to Query.columns.

Sort Object

The query.Sort object describes how query results are sorted (for example, ascending or descending, case sensitive or case insensitive, and so on).

To create a sort:

- Use Query.createSort(options) to create a sort on the initial query definition created with query.create(options).
- Use Component.createSort(options) to create a sort based on a join relationship created with Query.autoJoin(options), Query.joinFrom(options)/Query.joinTo(options), Component.autoJoin(options), or Component.joinFrom(options)/Component.joinTo(options).
- Assign all created sorts as array values to Query.sort.

ResultSet and Result Objects

When you are ready to execute your query, call Query.run(). This method returns a query.ResultSet object, which encapsulates the metadata for the set of results returned by the query.

To access your actual query results, iterate through the ResultSet.results array. Each member of the ResultSet.results array is a query.Result object. The query.Result object encapsulates a single row of the result set.

Page, PagedData, and PageRange Objects

You also can execute your query by calling Query.runPaged(). This method returns a query.PagedData object, which encapsulates a set of paged query results.

To access your query results, iterate through the paged query results using PagedData.iterator(). You can access each page of the query results, which are represented by query.Page objects. The query.PageRange object encapsulates the range of query results for a page.

N/query Module Terminology

Term	Definition	For More Information
Aggregate function	An aggregate function performs a calculation on a column of values and returns a single value. You can add aggregate functions to conditions and query results columns.	See query.Aggregate, Component.createColumn(options), Component.createCondition(options), Query.createColumn(options), and Query.createCondition(options).
Column	A column describes the field types (columns) that are displayed from the query results. A column is also known as a query results column.	See query.Column.
Component	When you script queries with the N/query module, your query is made up of one or more components, which are represented as query.Component objects. You can think of a component as a building block; each new component builds on the previous component created. ■ The first component created represents the initial search type and is a child of query.Query. ■ Each subsequent component created is a child of the previous component. ■ The last component created encapsulates the join relationship between it and all of its parent components. A query always contains at least one component: the root component. When you create the initial query definition using query.create(options), the root component is created automatically. Queries with joins contain multiple components. A new component is created each time you create a join using one of the following methods: ■ Query.autoJoin(options), Query.joinFrom(options), or Query.joinTo(options) ■ Component.autoJoin(options), Component.joinFrom(options), or Component.joinTo(options)	See query.Component.
Condition	A condition narrows the query results.	See query.Condition.

Term	Definition	For More Information
Formula	Formulas can be used to create conditions and columns.	See the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
Group	You can summarize your query results into unique groups of column values.	See Column.groupBy.
Join	A join lets you create a query based on a field type that is shared between two record types. You can use Query.autoJoin(options) and Component.autoJoin(options) to create a join relationship automatically based on a field that you specify. You can use Query.joinFrom(options)/Query.joinTo(options) and Component.joinFrom(options)/Component.joinTo(options) to create explicit directional join relationships from one component to another.	See query.Query and query.Component.
Page	A page represents one page from a set of paged query results. When you create a query with the N/query module, you can return the results as one result set or a set of paged results.	See Query.runPaged(), query.PagedData, query.PageRange, and query.Page.
Paged data	Paged data represents a set of paged query results.	See Query.runPaged(), query.PagedData, query.PageRange, and query.Page.
Page range	A page range is a set of pages from a set of paged query results.	See Query.runPaged(), query.PagedData, query.PageRange, and query.Page.
Result	A result is a single row from a result set.	See Query.run(), query.ResultSet and query.Result.
Result set	A result set is a set of query results.	See Query.run(), query.ResultSet and query.Result.
Query definition	The query definition is the initial search type you define, plus any subsequent joins you define. The initial query definition is created with query.create(options).	See query.Query.
Search type	The search type is the initial search type of your query definition. It represents the record type you want to search for. It is set with the query.Type enum during the execution of query.create(options). For example, if you want to search for customer records, specify `query.Type.CUSTOMER` as the search type when you call query.create(options).	See query.Query and query.Type.
Sort	A sort is placed on a query results column to describe how the query results are sorted (for example, ascending or descending, case sensitive or case insensitive, and so on).	See query.Sort, Query.createSort(options), and Component.createSort(options).

N/query Module Script Walkthrough

This topic walks through the two script examples shown under N/query Module Script Samples.

Example 1

```
require(['N/query'],
    function(query) {

        // Use query.create(options) to create your initial
        // query definition.
        var search = query.create({
            type: query.Type.CUSTOMER
        });

        // Use Query.autoJoin(options) to create your first join.
         var salesrep = search.autoJoin({
            fieldId: 'salesrep'
        });

        // Use Component.autoJoin(options) to create your second
        // join and each subsequent join.
         var location = salesrep.autoJoin({
            fieldId: 'location'
        });

        // Use Query.createCondition(options) to create
        // conditions for your initial query definition.
         var cond1 = search.createCondition({
            fieldId: 'id',
            operator: query.Operator.EQUAL,
            values: 107
        });
         var cond2 = search.createCondition({
            fieldId: 'id',
            operator: query.Operator.EQUAL,
            values: 2647
        });

        // Use Component.createCondition(options) to create
        // conditions for your joins
         var cond3 = salesrep.createCondition({
            fieldId: 'email',
            operator: query.Operator.START_WITH_NOT,
            values: 'foo'
        });

        // If you have one condition, assign it to the
        // Query.condition property.
        // If you have multiple conditions, logically
        // connect them with Query.and(), Query.or(),
        // and Query.not(). Then assign the statement to the
        // Query.condition property.
         search.condition = search.and(
            cond3, search.or(cond1, cond2)
        );

        // Use Query.createColumn(options) to create columns
        // for your initial query definition. Use
        // Component.createColumn(options) to create columns for
        // your joins. Assign each column, as an array member, to
        // the Query.columns property.
```

```
 search.columns = [
      search.createColumn({
        fieldId: 'entityid'
    }),
      search.createColumn({
        fieldId: 'id'
    }),
      salesrep.createColumn({
        fieldId: 'entityid'
    }),
      salesrep.createColumn({
        fieldId: 'email'
    }),
      salesrep.createColumn({
        fieldId: 'hiredate'
    }),
      location.createColumn({
        fieldId: 'name'
    })
];

// Use Query.createSort(options) to create an ascending or
// descending sort on columns created for your initial
// query definition. Assign each sort, as an array member,
// to the Query.sort property.
 search.sort = [
      search.createSort({
        column: search.columns[3]
    }),
      search.createSort({
        column: search.columns[0],
        ascending: false
    })
];

// Use Query.run() to synchronously execute your query
// and return the metadata for a set of results. You can use
// Query.promise.run() as an asynchronous alternative.
 var resultSet = search.run();

// The ResultSet.results property holds an array of your actual
// results. Each array member is a query.Result object. Iterate
// through the array to access the results.
 var results = resultSet.results;
 results.forEach(function(result) {
    log.debug(result.values);
});
 log.debug(resultSet.types);

 log.error(
    search.root === location.parent.parent
);
 log.error(
    search.root.child.salesrep === location.parent
);
```

```
         log.error(
            search.child.salesrep === location.parent
        );
         log.error(
            search.child.salesrep.child.location === location
        );
});
```

Example 2

```
require(['N/query'],
    function(query) {

        // Use query.create(options) to create your initial
        // query definition.
          var search = query.create({
            type: query.Type.TRANSACTION
        });

        // Use query.autoJoin(options) to create your first join.
          var entity = search.autoJoin({
            fieldId: 'entity'
        });

        // Use Query.createColumn(options) to create columns
        // for your initial query definition. Use
        // Component.createColumn(options) to create columns for
        // your joins. Assign each column, as an array member, to
        // the Query.columns property.
          search.columns = [
            entity.createColumn({
                fieldId: 'subsidiary'
            })
        ];

        // Use Query.createSort(options) to create an ascending or
        // descending sort on columns created for your initial
        // query definition. Assign each sort, as an array member,
        // to the Query.sort property.
          search.sort = [
            search.createSort({
                column: search.columns[0],
                ascending: false
            })
        ];

        // Use Query.runPaged() to synchronously execute your query
        // and return the metadata for an array of paged results. You can use
        // Query.promise.runPaged() as an asynchronous alternative.
          var results = search.runPaged({
            pageSize: 10
        });

        log.debug(results.pageRanges.length);
          log.debug(results.count);
        // Use one of the following ways to iterate through the array
        // to access the paged results.

        // First way to fetch results
          var iterator = results.iterator();
            iterator.each(function(result) {
                var page = result.value;
                log.debug(page.pageRange.size);
                return true;
          });

        // Second way to fetch results (you can also use a forEach loop)
          for (var i = 0; i < results.pageRanges.length; i++)  {
                var page = results.fetch(i);
                log.debug(page.pageRange.size);
          }
});
```

query.Column

Object Description	Encapsulates a query result column. The `query.Column` object is the equivalent of the search.Column object in the N/search Module. The `query.Column` object describes the field types (columns) that are displayed from the query results. To create columns: ■ Use Query.createColumn(options) to create a column on the initial query definition created with query.create(options). ■ Use Component.createColumn(options) to create a column on a join relationship created with Query.autoJoin(options) or Component.autoJoin(options). ■ Assign all created columns as array values to Query.columns. For an example, see Syntax.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Column Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
```

```
search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Column.aggregate

Property Description	Describes an aggregate function that is performed on the query result column. An aggregate function performs a calculation on the column values and returns a single value. This property is set when Query.createColumn(options) or Component.createColumn(options) is executed. For a list of supported aggregate functions, see the query.Aggregate enum.
Type	string (read-only)
Module	N/query Module
Parent Object	query.Column
Sibling Object Members	Column Object Members
Since	2018.1

Column.component

Property Description	Holds a reference to the query.Component object to which this query result column belongs. This property is set when Query.createColumn(options) or Component.createColumn(options) is executed.

Type	query.Component object (read-only)
Module	N/query Module
Parent Object	query.Column
Sibling Object Members	Column Object Members
Since	2018.1

Column.fieldId

Property Description	Holds the name of the query result column. This property is set during the execution of Query.createColumn(options) or Component.createColumn(options). This property and the Column.formula property cannot be set at the same time.
Type	string (read-only)
Module	N/query Module
Parent Object	query.Column
Sibling Object Members	Column Object Members
Since	2018.1

Column.formula

Property Description	Describes a formula used to create the query result column. This property is set during the execution of Query.createColumn(options) or Component.createColumn(options). This property and the Column.fieldId property cannot be set at the same time. For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
Type	string (read-only)
Module	N/query Module
Parent Object	query.Column
Sibling Object Members	Column Object Members
Since	2018.1

Column.groupBy

Property Description	Indicates whether the query results are grouped by this query result column. This property is set during the execution of Component.createColumn(options).
Type	boolean (read-only)
Module	N/query Module
Parent Object	query.Column

Sibling Object Members	Column Object Members
Since	2018.1

Column.type

Property Description	Describes the return type of the formula used to create the query result column. This property is set during the execution of Query.createColumn(options) or Component.createColumn(options). If a formula is specified when these methods are called, this property contains the return type of the formula. If a formula is not specified, this property is null. For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
Type	string (read-only)
Module	N/query Module
Parent Object	query.Column
Sibling Object Members	Column Object Members
Since	2018.1

query.Component

Object Description	Encapsulates one component of the query definition. Each new component is created as a child to the previous component. All components exist as children to the query definition (query.Query). You can think of a component as a building block; each new component builds on the previous component created. The last component created encapsulates the relationship between it and all of its parent components. The query definition always contains at least one component. Queries with joins contain multiple components. The query definition (query.Query) contains a child `query.Component` object for each of the following: ▪ **The initial query definition:** The initial `query.Component` object is called the root component. It encapsulates the initial search type passed to query.create(options). The root component is automatically created with the query.Query object and is a child of the query.Query object. The Query.root property contains a reference to the root component. ▪ **The first join:** The second `query.Component` object is created with Query.autoJoin(options). It encapsulates the relationship between the initial query definition and the second search type. This relationship is determined by the join ID passed to Query.autoJoin(options) . The second `query.Component` object is a child of the root component. ▪ **Each subsequent join:** The third `query.Component` object is created with Component.autoJoin(options). All subsequent joins and their respective `query.Component` objects are also created with Component.autoJoin(options) . Each of these `query.Component` objects encapsulates the relationship between all previous search types and the new search type. This relationship is determined by the join ID passed to Component.autoJoin(options).
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Component Object Members

Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Component.autoJoin(options)

Method Description	Creates a join relationship. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use Query.autoJoin(options) to create your first join (query.Component). Then use `Component.autoJoin(options)` to create each subsequent join (query.Component).

	Important: For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.

Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.autoJoin({
    fieldId: 'entity'
});
```

```
search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

Component.createColumn(options)

Method Description	Creates a query result column based on the query.Component object. The query.Column object is the equivalent of the search.Column object in the N/search Module. The query.Column object describes the field types (columns) that are displayed from the query results. To create columns: ■ Use `Component.createColumn(options)` to create conditions on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). Use this method in one of two ways: □ Pass in an argument for the parameter `options.fieldId`. □ Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`. ■ If needed, use Query.createColumn(options) to create columns on the initial query definition created with query.create(options). ■ Assign all created columns as array values to Query.columns. For an example, see Syntax.
Returns	query.Column object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required if `options.formula` is not used	The name of the query result column. This value sets the Column.fieldId property. Obtain this value from the Records Browser:

Parameter	Type	Required / Optional	Description
			1. Go to the appropriate record type. 2. Scroll until you see the Search Columns table. 3. Locate the appropriate value in the Internal ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.formula	string	required if `options.fieldId` is not used	The formula used to create the query result column. This value sets the Column.formula property. For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
options.type	string	optional if `options.formula` is used	If you use the `options.formula` parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Column.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.aggregate	string	optional	Use this parameter to run an aggregate function on your query result column. An aggregate function performs a calculation on the column values and returns a single value. This value sets the Column.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.groupBy	boolean	optional	Indicates whether the query results are grouped by this query result column. This value sets the Column.groupBy property. If you do not pass in an argument, the default value is set to `false`.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Component.createCondition(options)

Method Description	Creates a condition (query filter) based on the query.Component object. A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects. To create conditions: ■ Use `Component.createCondition(options)` to create conditions on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). Use this method in one of two ways: □ Pass in arguments for the parameters `options.fieldId`, `options.operator`, and `options.values`. The combination of these arguments translates to *<filter column><operator><field value>* (for example, 'city' equals 'Boston'). □ Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`. ■ If needed, use Query.createCondition(options) to create conditions on the initial query definition created with query.create(options). ■ If you have multiple conditions, use them to create a new nested condition with the methods Query.and(), Query.or(), and Query.not(). ■ Assign your simple or nested condition to Query.condition. For an example, see Syntax.
Returns	query.Condition object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component

Sibling Object Members	Component Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required if `options.operator` and `options.values` are used	The name of the condition. This value sets the Condition.fieldId property. Obtain this value from the Records Browser: 1. Go to the appropriate record type. 2. Scroll until you see the Search Filters table. 3. Locate the appropriate value in the Internal ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.operator	string	required if `options.fieldId` and `options.values` are used	The operator used by the condition. This value sets the Condition.operator parameter. Use the appropriate query.Operator enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.values	string[]	required if `options.fieldId` and `options.operator` are used	An array of string values. This value sets the Condition.values property.
options.formula	string	required if `options.fieldId`, `options.operator`, and `options.values` are **not** used	The formula used to create the condition. This value sets the Condition.formula property. For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
options.type	string	optional if `options.formula` is used	If you use the `options.formula` parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Condition.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.aggregate	string	optional	Use this parameter to run an aggregate function on a condition. An aggregate function performs a calculation on the condition values and returns a single value. This value sets the Condition.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Component.createSort(options)

Method Description	Creates a sort based on the query.Component object. The query.Sort object describes a sort that is placed on a particular query result column or condition. To create a sort: ■ Use `Component.createSort(options)` to create a sort based on a join relationship created with Query.autoJoin(options) or Component.autoJoin(options). ■ Use Query.createSort(options) to create a sort based on the initial query definition created with query.create(options). ■ Assign all created sorts as array values to Query.sort. For an example, see Syntax.
Returns	query.Sort

Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.column	query.Column	required	The query result column that you want to sort by. This value sets the Sort.column property.
options.ascending	boolean	optional	Indicates whether the sort direction is ascending. This value sets the Sort.ascending property. The default value of this property is `true`, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to `false`.
options.caseSensitive	boolean	optional	Indicates whether the sort is case sensitive. This value sets the Sort.caseSensitive property. If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same. For example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending: ▪ Banana ▪ Orange ▪ apple ▪ grapefruit ▪ kiwi Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending: ▪ apple ▪ Banana ▪ grapefruit ▪ kiwi ▪ Orange The default value of this property is `false`.
options.locale	string	optional	The locale to use for the sort. This value sets the Sort.locale property. A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted. For example, languages that share

Parameter	Type	Required / Optional	Description
			the same alphabet may sort characters differently. Use this property to ensure that query results are sorted using locale-specific rules. Use the appropriate query.SortLocale enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.nullsLast	boolean	optional	Indicates whether query results with null values are listed at the end of the query results. This value sets the Sort.nullsLast property. The default value of this property is the value of the `options.ascending` property. For example, if the `options.ascending` property is set to `true`, the `options.nullsLast` property is also set to `true`.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Component.join(options)

Method Description	Creates a join relationship. This method is an alias to Component.autoJoin(options). Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use Query.autoJoin(options) to create your first join (query.Component). Then use `Component.join(options)` to create each subsequent join (query.Component). **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. This value determines the columns on which the components are joined and the type of the newly joined component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

Component.joinFrom(options)

Method Description	Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the Component.source property on the returned query.Component object. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use this method to create explicit directional joins from other components to this component. **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.source	string	required	The search type of the component joined to this component. This value sets the Component.source property. This value can be described as the inverse relationship of this component, and it determines the source search type of the newly joined component.

Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.EMPLOYEE
});

var salesorder = search.joinFrom({
    fieldId: 'salesrep',
    source: 'salesorder'
});

var items = salesorder.autoJoin({
    fieldId: 'item'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'hiredate'
    }),
    salesorder.createColumn({
        fieldId: 'id'
    }),
    salesorder.createColumn({
        fieldId: 'trandate'
    })
];

var sort1 = search.createSort({
    column: search.columns[0],
    ascending:false
});
var sort2 = search.createSort({
    column: search.columns[1],
    ascending:true
});
search.sort = [sort1, sort2];

var results = search.run();
```

Component.joinTo(options)

Method Description	Creates an explicit directional join relationship to another component from this component (a polymorphic join). This method sets the Component.target property on the returned query.Component object. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use this method to create explicit directional joins to other components from this component. **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component.

Parameter	Type	Required / Optional	Description
			Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.target	string	required	The search type of the component joined to this component. This value sets the Component.target property. This value can be described as the polymorphic relationship of this component, and it determines the target search type of the newly joined component.

Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.joinTo({
    fieldId: 'entity',
    target: query.Type.CUSTOMER
});

search.columns = [
    entity.createColumn({
        fieldId: 'subsidiary'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false
    })
];

var results = search.runPaged({
    pageSize: 10
});
```

Component.child

Property Description	Holds a references to children of this component. The value of this property is an object of key/value pairs. Each key is the name of a child component. Each respective value refers to the corresponding query.Component object. The object values are set during the execution of Query.autoJoin(options) and Component.autoJoin(options). The order of the key/value pairs reflects the parent/child hierarchy.
Type	Object (read-only)
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Component.parent

Property Description	Holds a references to the parent query.Component object of this component. This property is set during the execution of Query.autoJoin(options) or Component.autoJoin(options).
Type	string (read-only)
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Component.source

Property Description	Describes the search type of the component joined to this component. This property can also be described as the inverse relationship of this component. This property is set during the execution of Query.joinFrom(options) and Component.joinFrom(options).
Type	string (read-only)
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Component.target

Property Description	Describes the search type of this component. This property can also be described as the polymorphic relationship of this component.

	This property is set during the execution of Query.joinTo(options) and Component.joinTo(options).
Type	string (read-only)
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

Component.type

Property Description	Describes the search type of this component. This property is set during the execution of Query.autoJoin(options) and Component.autoJoin(options).
Type	string (read-only)
Module	N/query Module
Parent Object	query.Component
Sibling Object Members	Component Object Members
Since	2018.1

query.Condition

Object Description	A condition narrows the query results. The `query.Condition` object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that `query.Condition` objects can contain other `query.Condition` objects. To create conditions: ▪ Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options). ▪ Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ▪ If you have multiple conditions, use them to create a new nested condition with the methods Query.and(), Query.or(), and Query.not(). ▪ Assign your simple or nested condition to Query.condition. For an example, see Syntax.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Condition Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Condition.aggregate

Property Description	Describes an aggregate function that is performed on the condition. An aggregate function performs a calculation on the condition values and returns a single value. This property is set during the execution of Query.createCondition(options) or Component.createCondition(options). **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string (read-only)
Module	N/query Module

Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.children

Property Description	Holds an array of child conditions used to create the parent condition. **Note:** This property is applicable to only parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	query.Condition[]
Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.component

Property Description	Describes the component used to created the condition This property is set during the execution of Query.createCondition(options) and Component.createCondition(options). **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string (read-only)
Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.fieldId

Property Description	Holds the name of the condition. This property is set during the execution of Query.createCondition(options) and Component.createCondition(options). **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string (read-only)

Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.formula

Property Description	Describes the formula used to create the condition. This property is set during the execution of Query.createCondition(options) and Component.createCondition(options). For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips. **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string (read-only)
Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.operator

Property Description	Holds the name of the operator used to create the condition. This property is set during the execution of Query.createCondition(options) and Component.createCondition(options). **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string (read-only)
Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.type

Property Description	The return type of the formula used to create the condition. This property is set during the execution of Query.createCondition(options) or Component.createCondition(options).

	For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips. **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string (read-only)
Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

Condition.values

Property Description	Holds an array of values used by an operator to create the condition. This property is set by passing in values for `options.fieldId`, `options.operator` and `options.values` during the execution of Query.createCondition(options) or Component.createCondition(options). **Note:** This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().
Type	string[] (read-only)
Module	N/query Module
Parent Object	query.Condition
Sibling Object Members	Condition Object Members
Since	2018.1

query.Page

Object Description	One page of the paged query results.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Page Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var results = search.runPaged({
    pageSize: 10});

log.debug(results.pageRanges.length);
log.debug(results.count);

//First way to fetch results
var iterator = results.iterator();
iterator.each(function(result) {
    var page = result.value;
        log.debug(page.pageRange.size);
        return true;
})

//Second way to fetch results
for (var i = 0; i < results.pageRanges.length; i++)  {
    var page = results.fetch(i);
    log.debug(page.pageRange.size);
}
```

Page.data

Property Description	References the query results contained in this page.
Type	query.ResultSet (read-only)
Module	N/query Module
Parent Object	query.Page
Sibling Object Members	Page Object Members
Since	2018.1

Page.isFirst

Property Description	Indicates whether the page is the first of the paged query results.
Type	boolean (read-only)
Module	N/query Module
Parent Object	query.Page
Sibling Object Members	Page Object Members
Since	2018.1

Page.isLast

Property Description	Indicates whether the page is the last of the paged query results.
Type	boolean (read-only)
Module	N/query Module
Parent Object	query.Page

Sibling Object Members	Page Object Members
Since	2018.1

Page.pageRange

Property Description	The range of query results for this page.
Type	query.PageRange (read-only)
Module	N/query Module
Parent Object	query.Page
Sibling Object Members	Page Object Members
Since	2018.1

Page.pagedData

Property Description	References the set of paged query results that this page is from.
Type	query.PagedData (read-only)
Module	N/query Module
Parent Object	query.Page
Sibling Object Members	Page Object Members
Since	2018.1

query.PagedData

Object Description	Encapsulates a set of paged query results. This object also contains information about the set of paged results it encapsulates. Use Query.runPaged() or Query.runPaged.promise() to create this object.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	PagedData Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var results = search.runPaged({
    pageSize: 10});
```

```
log.debug(results.pageRanges.length);
log.debug(results.count);

//First way to fetch results
var iterator = results.iterator();
iterator.each(function(result) {
    var page = result.value;
        log.debug(page.pageRange.size);
        return true;
})

//Second way to fetch results
for (var i = 0; i < results.pageRanges.length; i++)  {
    var page = results.fetch(i);
    log.debug(page.pageRange.size);
}
```

PagedData.iterator()

Method Description	Standard SuiteScript 2.0 object for iterating through results
Returns	Iterator object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.PagedData
Sibling Object Members	PagedData Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var results = search.runPaged({
    pageSize: 10});

log.debug(results.pageRanges.length);
log.debug(results.count);

//First way to fetch results
var iterator = results.iterator();
iterator.each(function(result) {
    var page = result.value;
        log.debug(page.pageRange.size);
        return true;
})

//Second way to fetch results
```

```
for (var i = 0; i < results.pageRanges.length; i++)  {
    var page = results.fetch(i);
    log.debug(page.pageRange.size);
}
```

PagedData.count

Property Description	Describes the total number of paged query result rows.
Type	number (read-only)
Module	N/query Module
Parent Object	query.PagedData
Sibling Object Members	PagedData Object Members
Since	2018.1

PagedData.pageRanges

Property Description	Holds an array of page ranges for the paged query results.
Type	query.PageRange[]
Module	N/query Module
Parent Object	query.PagedData
Sibling Object Members	PagedData Object Members
Since	2018.1

PagedData.pageSize

Property Description	Describes the number of query result rows per page.
Type	number (read-only)
Module	N/query Module
Parent Object	query.PagedData
Sibling Object Members	PagedData Object Members
Since	2018.1

query.PageRange

Object Description	Encapsulates the range of query results for a page.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	PageRange Object Members

Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var results = search.runPaged({
    pageSize: 10});

log.debug(results.pageRanges.length);
log.debug(results.count);

//First way to fetch results
var iterator = results.iterator();
iterator.each(function(result) {
    var page = result.value;
        log.debug(page.pageRange.size);
        return true;
})

//Second way to fetch results
for (var i = 0; i < results.pageRanges.length; i++)  {
    var page = results.fetch(i);
    log.debug(page.pageRange.size);
}
```

PageRange.index

Property Description	Describes the array index for this page range.
Type	number (read-only)
Module	N/query Module
Parent Object	query.PageRange
Sibling Object Members	PageRange Object Members
Since	2018.1

PageRange.size

Property Description	Describes the number of query result rows in this page range.
Type	number (read-only)
Module	N/query Module
Parent Object	query.PageRange
Sibling Object Members	PageRange Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var results = search.runPaged({
    pageSize: 10});

log.debug(results.pageRanges.length);
log.debug(results.count);

//First way to fetch results
var iterator = results.iterator();
iterator.each(function(result) {
    var page = result.value;
        log.debug(page.pageRange.size);
        return true;
})

//Second way to fetch results
for (var i = 0; i < results.pageRanges.length; i++)  {
    var page = results.fetch(i);
    log.debug(page.pageRange.size);
}
```

query.Query

Object Description	The `query.Query` object encapsulates the query definition. To create a query with the N/query module: 1. Use the query.create(options) method to create your query definition (this object). The initial query definition uses one search type. For available search types, see query.Type. 2. After you create the initial query definition, use Query.autoJoin(options) to create your first join. 3. Then use Component.autoJoin(options) to create all subsequent joins. The query definition always contains at least one query.Component object. The query.Component object encapsulates one component of the query definition. Each new component is created as a child to the previous component, and all components exist as children to the query definition. You can think of a component as a building block; each new component builds on the previous component created. The last component created encapsulates the relationship between it and all of its parent components. Queries with joins contain multiple components. The query definition contains a child query.Component object for each of the following: ▪ **The initial query definition:** The initial query.Component object is called the root component. It encapsulates the initial search type passed to query.create(options). The root component is automatically created with the initial query definition and is a child to the `query.Query` object. The Query.root property contains a reference to the root component. ▪ **The first join:** The second query.Component object is created with Query.autoJoin(options). It encapsulates the relationship between the initial query definition and the second search type. This relationship is determined by the join ID passed to Query.autoJoin(options). The second query.Component object is a child to the root component. ▪ **Each subsequent join:** The third query.Component object is created with Component.autoJoin(options). All subsequent joins are also created with

	Component.autoJoin(options) . Each of these query.Component objects encapsulates the relationship between all previous search types and the new search type. This relationship is determined by the join ID passed to Component.autoJoin(options).
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Query Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

Query.and()

Method Description	Creates a new condition (a query.Condition object) that corresponds to a logical conjunction (AND) of the arguments passed to the method. The arguments must be one or more query.Condition objects. A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects. To create conditions: ■ Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options). ■ Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options) and Component.autoJoin(options).

	■ If you have multiple conditions, use them to create a new parent condition with the methods `Query.and()`, Query.or(), and Query.not(). ■ Assign your parent condition to Query.condition. For an example, see Syntax.
Returns	query.Condition object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Parameter	Type	Required / Optional	Description
condition 1 — n	query.Condition	Required	One or more condition objects. There is no limit on the number of conditions you can specify.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
```

```
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Query.autoJoin(options)

Method Description	Creates a join relationship. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use `Query.autoJoin(options)` to create your first join (query.Component). Then use Component.autoJoin(options) to create each subsequent join (query.Component). **Note:** This method is a shortcut for the chained Query.root and Component.autoJoin(options): `Query.root.join(options)`. The Query.root property references the root component, which is a query.Component object. **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. This value determines the columns on which the components are joined and the type of the newly joined component.

Parameter	Type	Required / Optional	Description
			Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.

Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.autoJoin({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

Query.createColumn(options)

Method Description	This method creates a query result column based on the query.Query object. The query.Column object is the equivalent of the search.Column object in the N/search Module. The query.Column object describes the field types (columns) that are displayed from the query results. To create columns: ▪ Use `Query.createColumn(options)` to create conditions on the initial query definition created with query.create(options). Use this method in one of two ways: ▫ Pass in an argument for the parameter `options.fieldId`.

	◦ Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`. ■ If needed, use Component.createColumn(options) to create conditions on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ■ Assign all created columns as array values to Query.columns. For an example, see Syntax. **Note:** This method is a shortcut for the chained Query.root and Component.createColumn(options): `Query.root.createColumn(options)`. The Query.root property references the root component, which is a query.Component object.
Returns	query.Column object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required if `options.formula` is not used	The name of the query result column. This value sets the Column.fieldId property. Obtain this value from the Records Browser: 1. Go to the appropriate record type. 2. Scroll until you see the Search Columns table. 3. Locate the appropriate value in the Internal ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.formula	string	required if `options.fieldId` is not used	The formula used to create the query result column. This value sets the Column.formula property. For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
options.type	string	optional if `options.formula` is used	If you use the `options.formula` parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Column.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.

Parameter	Type	Required / Optional	Description
options.aggregate	string	optional	Use this parameter to run an aggregate function on your query result column. An aggregate function performs a calculation on the column values and returns a single value. This value sets the Column.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.groupBy	boolean	optional	Indicates whether the query results are grouped by this query result column. This value sets the Column.groupBy property. If you do not pass in an argument, the default value is set to `false`.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];
```

```
var resultSet = search.run();
```

Query.createCondition(options)

Method Description	This method creates a condition (query filter) based on the query.Query object. A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects. To create conditions: ■ Use `Query.createCondition(options)` to create conditions on the initial query definition created with query.create(options). Use this method in one of two ways: □ Pass in arguments for the parameters `options.fieldId`, `options.operator`, and `options.values`. The combination of these arguments translates to *<filter column><operator><field value>* (for example, 'city' equals 'Boston'). □ Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`. ■ If needed, use Component.createCondition(options) to create conditions on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ■ If you have multiple conditions, use them to create a new nested condition with the methods Query.and(), Query.or(), and Query.not(). ■ Assign your simple or nested condition to Query.condition. For an example, see Syntax. **Note:** This method is a shortcut for the chained Query.root and Component.createCondition(options): `Query.root.createCondition(options)`. The Query.root property references the root component, which is a query.Component object.
Returns	query.Condition object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required if `options.operator` and `options.values` are used	The name of the condition. This value sets the Condition.fieldId property. Obtain this value from the Records Browser:

Parameter	Type	Required / Optional	Description
			1. Go to the appropriate record type. 2. Scroll until you see the Search Filters table. 3. Locate the appropriate value in the Internal ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.operator	string	required if `options.fieldId` and `options.values` are used	The operator used by the condition. This value sets the Condition.operator parameter. Use the appropriate query.Operator enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.values	string[]	required if `options.fieldId` and `options.operator` are used	An array of string values. This value sets the Condition.values property.
options.formula	string	required if `options.fieldId`, `options.operator`, and `options.values` are **not** used	The formula used to create the condition. This value sets the Condition.formula property. For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips.
options.type	string	optional if `options.formula` is used	If you use the `options.formula` parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Condition.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.aggregate	string	optional	Use this parameter to run an aggregate function on a condition. An aggregate function performs a calculation on the condition values and returns a single value. This value sets the Condition.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});
```

```
var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Query.createSort(options)

Method Description	This method creates a sort based on the query.Query object. The query.Sort object describes a sort that is placed on a particular query result column. To create a sort: ■ Use `Search.createSort(options)` to create a sort based on the initial query definition created with query.create(options). ■ Use Component.createSort(options) to create a sort based on a join relationship created with Query.autoJoin(options) or Component.autoJoin(options). ■ Assign all created sorts as array values to Query.sort. For an example, see Syntax. **Note:** This method is a shortcut for the chained Query.root and Component.createSort(options): `Query.root.createSort(options)`. The Query.root property references the root component, which is a query.Component object.
Returns	query.Sort object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.column	query.Column	required	The query result column that you want to sort by. This value sets the Sort.column property.
options.ascending	boolean	optional	Indicates whether the sort direction is ascending. This value sets the Sort.ascending property. The default value of this property is `true`, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to `false`.
options.caseSensitive	boolean	optional	Indicates whether the sort is case sensitive. This value sets the Sort.caseSensitive property. If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same. For example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending: ▪ Banana ▪ Orange ▪ apple ▪ grapefruit ▪ kiwi Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending: ▪ apple ▪ Banana ▪ grapefruit ▪ kiwi ▪ Orange The default value of this property is `false`.
options.locale	string	optional	The locale to use for the sort. This value sets the Sort.locale property. A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted. For example, languages that share the same alphabet may sort characters differently. Use this property to ensure that query results are sorted using locale-specific rules. Use the appropriate query.SortLocale enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.nullsLast	boolean	optional	Indicates whether query results with null values are listed at the end of the query results. This value sets the Sort.nullsLast property. The default value of this property is the value of the `options.ascending` property. For example, if the

Parameter	Type	Required / Optional	Description
			`options.ascending` property is set to `true`, the `options.nullsLast` property is also set to `true`.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Query.join(options)

Method Description	Creates a join relationship.

	Important: This method is an alias to Query.autoJoin(options). Use Query.autoJoin(options) instead of this method to create simple joins. Use Query.joinFrom(options) and Query.joinTo(options) to create explicit directional joins. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use `Query.join(options)` to create your first join (query.Component). Then use Component.autoJoin(options) to create each subsequent join (query.Component). **Note:** This method is a shortcut for the chained Query.root and Component.join(options): `Query.root.join(options)`. The Query.root property references the root component, which is a query.Component object. **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. This value determines the columns on which the components are joined and the type of the newly joined component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

Query.joinFrom(options)

Method Description	Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the Component.source property on the returned query.Component object. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use this method to create your first join as an explicit directional join from another component to this component. **Note:** This method is a shortcut for the chained Query.root and Component.joinFrom(options): `Query.root.joinFrom(options)`. The Query.root property references the root component, which is a query.Component object. **Important:** For the 2018.2 beta release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query

Sibling Object Members	Query Object Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.source	string	required	The search type of the component joined to this component. This value sets the Component.source property. This value can be described as the inverse relationship of this component, and it determines the source search type of the newly joined component.

Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

Query.joinTo(options)

Method Description	Creates an explicit directional join relationship to another component from this component (a polymorphic join). This method sets the Component.target property on the returned query.Component object. Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use this method to create your first join as an explicit directional join to another component from this component. **Note:** This method is a shortcut for the chained Query.root and Component.joinTo(options): `Query.root.autoJoin(options)`. The Query.root property references the root component, which is a query.Component object. **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.
Returns	query.Component object

Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.fieldId	string	required	The column type (field type) that joins the parent component to the new component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.
options.target	string	required	The search type of the component joined to this component. This value sets the Component.target property. This value can be described as the polymorphic relationship of this component, and it determines the target search type of the newly joined component.

Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

Query.run()

Method Description	Executes the query and returns the query result set.
Returns	query.ResultSet
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	10 Usage Units
Module	N/query Module
Parent Object	query.Query

Sibling Object Members	Query Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Query.run.promise()

Method Description	Executes the query asynchronously and returns the query result set.
Returns	query.ResultSet

Supported Script Types	Client scripts For more information, see SuiteScript 2.0 Script Types.
Governance	10 Usage Units
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Query.runPaged()

Method Description	Executes the query and returns a set of paged results.
Returns	query.PagedData
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	10 Usage Units
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    name: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

```
// Use the count property to count the
// search results easily
var resultCount = search.runPaged({
    pageSize: 10
}).count;
```

Query.runPaged.promise()

Method Description	Executes the query asynchronously and returns a set of paged results.
Returns	query.PagedData
Supported Script Types	Client scripts For more information, see SuiteScript 2.0 Script Types.
Governance	10 Usage Units
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Query.not()

Method Description	Creates a new condition (a query.Condition object) that corresponds to a logical negation (NOT) of the argument passed to the method. The argument must be a query.Condition object. A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects. To create conditions: ▪ Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options). ▪ Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ▪ If you have multiple conditions, use them to create a new parent condition with the methods Query.and(), Query.or(), and `Query.not()`. ▪ Assign your parent condition to Query.condition. For an example, see Syntax.
Returns	query.Condition
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Parameter	Type	Required / Optional	Description
condition	query.Condition	Required	One condition object.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Query.or()

Method Description	Creates a new condition (a query.Condition object) that corresponds to a logical disjunction (OR) of the arguments passed to the method. The arguments must be one or more query.Condition objects. A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects. To create conditions:

	■ Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options). ■ Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ■ If you have multiple conditions, use them to create a new parent condition with the methods Query.and(), `Query.or()`, and Query.not(). ■ Assign your parent condition to Query.condition. For an example, see Syntax.
Returns	query.Condition object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Parameters

Parameter	Type	Required / Optional	Description
condition 1 — n	query.Condition	Required	One or more condition objects. There is no limit on the number of conditions you can specify.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Query.child

Property Description	Holds a references to children of this component. The value of this property is an object of key/value pairs. Each key is the name of a child component. Each respective value is the corresponding query.Component object. The object values are set with the execution of Query.autoJoin(options) and Component.autoJoin(options). The order of the key/value pairs reflects the parent/child hierarchy.
Type	Object
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Query.columns

Property Description	Holds an array of result columns (query.Column objects) returned from the query. The query.Column object is the equivalent of the search.Column object in the N/search Module. The query.Column object describes a field type (column) that is returned from the query results. To create columns: ■ Use Query.createColumn(options) to create conditions on the initial query definition created with query.create(options). ■ Use Component.createColumn(options) to create conditions on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ■ Assign all created columns as array values to `Query.columns`. For an example, see Syntax.
Type	query.Column[]
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members

Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Query.condition

Property Description	References the simple or nested condition (a query.Condition object) that narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects. To create conditions:

	■ Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options). ■ Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). ■ If you have multiple conditions, use them to create a new nested condition with the methods Query.and(), Query.or(), and Query.not(). ■ Assign your simple or nested condition to `Query.condition`. For an example, see Syntax.
Type	query.Condition object
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Query.id

Property Description	Holds the ID of the query definition. This property has a value only for existing queries that are loaded using query.load(options). If you create a query using query.create(options) but do not save it, this property is null. **Important:** In the 2018.2 release, you can use the N/query module to load and delete existing searches, but you cannot save searches. You can save searches using the SuiteAnalytics Workbook UI.
Type	number (read-only)
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Query.name

Property Description	Holds the name of the query definition. This property has a value only for existing queries that are loaded using query.load(options). If you create a query using query.create(options) but do not save it, this property is null. **Important:** In the 2018.2 release, you can use the N/query module to load and delete existing searches, but you cannot save searches. You can save searches using the SuiteAnalytics Workbook UI.
Type	string (read-only)
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Query.root

Property Description	References the root component of the query definition. The initial query.Component object is called the root component. It encapsulates the initial search type passed to query.create(options). The root component is automatically created with the query.Query object and is a child of the query.Query object.
Type	query.Component (read-only)
Module	N/query Module
Parent Object	query.Query

Sibling Object Members	Query Object Members
Since	2018.1

Query.sort

Property Description	Holds an array of query result columns (query.Column objects) used for sorting. This object encapsulates a sort based on the query.Query or query.Component object. The query.Sort object describes a sort that is placed on a particular query result column. To create a sort: ▪ Use Query.createSort(options) to create a sort based on the initial query definition created with query.create(options). ▪ Use Component.createSort(options) to create a sort based on a join relationship created with Query.autoJoin(options) or Component.autoJoin(options). ▪ Assign all created sorts as array values to Query.sort. For an example, see Syntax.
Type	query.Sort[]
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Query.type

Property Description	Describes the initial search type of the query definition. This property is set during the execution of query.create(options).
Type	string (read-only)
Module	N/query Module
Parent Object	query.Query
Sibling Object Members	Query Object Members
Since	2018.1

query.Result

Object Description	Encapsulates a single row of the result set (query.ResultSet).
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Result Object Members
Since	2018.1

Result.columns

Property Description	Holds an array of query return column references. These array values are equivalent to the array values in ResultSet.columns.
Type	query.Column[] (read-only)
Module	N/query Module
Parent Object	query.Result
Sibling Object Members	Result Object Members
Since	2018.1

Result.values

Property Description	Describes the result values. Value types correspond to the ResultSet.types property. Array values correspond to the array values for ResultSet.columns and Result.columns.
Type	string[] or number[] or boolean[] (read-only)
Module	N/query Module
Parent Object	query.Result
Sibling Object Members	Result Object Members
Since	2018.1

query.ResultSet

Object Description	Encapsulates the set of results returned by the query. Use Query.run() or Query.run.promise() to create this object.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	ResultSet Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var resultSet = search.run();
var results = resultSet.results;
for (var i = results.length - 1; i >== 0; i--)
    log.debug(results[i].values);
```

ResultSet.iterator()

Method Description	Standard SuiteScript 2.0 object for iterating through results
Returns	Iterator object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Parent Object	query.ResultSet
Sibling Object Members	ResultSet Object Members

Since	2018.1

ResultSet.columns

Property Description	Holds an array of query return column references. The `ResultSet.columns` array values correspond with the ResultSet.types array values.
Type	query.Column[] (read-only)
Module	N/query Module
Parent Object	query.ResultSet
Sibling Object Members	ResultSet Object Members
Since	2018.1

ResultSet.results

Property Description	Holds an array of query.Result objects.
Type	query.Result[] (read-only)
Module	N/query Module
Parent Object	query.ResultSet
Sibling Object Members	ResultSet Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var resultSet = search.run();
var results = resultSet.results;
for (var i = results.length - 1; i >== 0; i--)
    log.debug(results[i].values);
```

ResultSet.types

Property Description	Holds an array of the return types for ResultSet.results. The `ResultSet.types` array values correspond with the ResultSet.columns array values.
Type	string[] (read-only)
Module	N/query Module
Parent Object	query.ResultSet
Sibling Object Members	ResultSet Object Members
Since	2018.1

query.Sort

Object Description	Encapsulates a sort based on the query.Query or query.Component object. The `query.Sort` object describes a sort that is placed on a particular query result column. To create a sort: ■ Use Query.createSort(options) to create a sort based on the initial query definition created with query.create(options). ■ Use Component.createSort(options) to create a sort based on a join relationship created with Query.autoJoin(options) or Component.autoJoin(options). ■ Assign all created sorts as array values to Query.sort. For an example, see Syntax.
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Module	N/query Module
Methods and Properties	Sort Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

Sort.ascending

Property Description	Indicates whether the sort direction is ascending. This property is set during the execution of Query.createSort(options) and Component.createSort(options). The default value of this property is `true`, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to `false`.
Type	boolean
Module	N/query Module
Parent Object	query.Sort
Sibling Object Members	Sort Object Members
Since	2018.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

Sort.caseSensitive

Property Description	Indicates whether the sort is case sensitive. This property is set during the execution of Query.createSort(options) and Component.createSort(options). If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same. For example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending: ■ Banana ■ Orange ■ apple ■ grapefruit ■ kiwi Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending: ■ apple ■ Banana ■ grapefruit ■ kiwi ■ Orange The default value of this property is `false`.
Type	boolean
Module	N/query Module
Parent Object	query.Sort
Sibling Object Members	Sort Object Members
Since	2018.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

Sort.column

Property Description	Describes the query result column that the query results are sorted by. This property is set during the execution of Query.createSort(options) and Component.createSort(options).
Type	query.Column (read-only)
Module	N/query Module
Parent Object	query.Sort
Sibling Object Members	Sort Object Members
Since	2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

Sort.locale

Property Description	The locale to use for the sort. This property uses values from the query.SortLocale enum. This property is set during the execution of Query.createSort(options) and Component.createSort(options). A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted. For example, languages that share the same alphabet may sort characters differently. Use this property to ensure that query results are sorted using locale-specific rules.
Type	string
Module	N/query Module
Parent Object	query.Sort
Sibling Object Members	Sort Object Members
Since	2018.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

Sort.nullsLast

Property Description	Indicates whether query results with null values are listed at the end of the query results. This property is set during the execution of Query.createSort(options) and Component.createSort(options). The default value of this property is the value of the Sort.ascending property. For example, if the Sort.ascending property is set to `true`, the `Sort.nullsLast` property is also set to `true`.
Type	boolean

Module	N/query Module
Parent Object	query.Sort
Sibling Object Members	Sort Object Members
Since	2018.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

query.create(options)

Method Description	Creates a query.Query object. Use this method to create your initial query definition. The initial query definition uses one search type. For available search types, see query.Type. After you create the initial query definition, use Query.autoJoin(options) to create your first join. Then use Component.autoJoin(options) to create all subsequent joins.
Returns	query.Query object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	None
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.type	string	required	The search type that you want to use for the initial query definition. Use the query.Type enum to set this value (for an example, see Syntax). When you execute `query.create(options)`, the Query.type property is set based on this value. **Important:** For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

query.delete(options)

Method Description	Deletes an existing query. Use this method to delete a query definition that was previously created using the SuiteAnalytics Workbook UI. After the query is deleted, it is no longer available and cannot be modified or executed. **Important:** In the 2018.2 release, you can use the N/query module to load and delete existing searches, but you cannot save searches. You can save searches using the SuiteAnalytics Workbook UI.
Returns	void
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	5 Usage Units
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.id	number	required	The ID of the query to delete.

Errors

Error Code	Thrown If
UNABLE_TO_DELETE_QUERY	A query with the specified ID cannot be deleted because the query does not exist or you do not have permission to delete it.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var deletedSearch = query.delete({
    id: 237
});
```

query.load(options)

Method Description	Loads an existing query as a query.Query object. Use this method to load a query definition that was previously created using the SuiteAnalytics Workbook UI. After the query is loaded, you can modify the query definition (for example, by setting additional property values), join the query definition with other search types, and execute the query in the same way as queries that you create using query.create(options). **Important:** In the 2018.2 release, you can use the N/query module to load and delete existing searches, but you cannot save searches. You can save searches using the SuiteAnalytics Workbook UI.
Returns	query.Query object
Supported Script Types	Client and server-side scripts For more information, see SuiteScript 2.0 Script Types.
Governance	5 Usage Units
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.id	number	required	The ID of the query to load.

Errors

Error Code	Thrown If
UNABLE_TO_LOAD_QUERY	A query with the specified ID cannot be loaded because the query does not exist or you do not have permission to load it.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.load({
    id: 237
});

var salesrep = search.autoJoin({
    fieldId: 'salesrep'
});

var resultSet = search.run();
```

query.Aggregate

Enum Description	Holds the string values for aggregate functions supported with the N/query Module. An aggregate function performs a calculation on the column or condition values and returns a single value. Each value in this enum (except `MEDIAN`) has two variants: distinct (using the `_DISTINCT` suffix) and nondistinct (using no suffix). The variant determines whether the aggregate function operates on all instances of duplicate values or on just a single instance of the value. For example, consider a situation in which the `MAXIMUM` aggregate function is used to determine the maximum of a set of values. When using the distinct variant (`MAXIMUM_DISTINCT`), the aggregate function considers each instance of duplicate values. So if the set of values includes three distinct values that are all equal and all represent the maximum value in the set, the aggregate function lists all three instances. When using the nondistinct variant (`MAXIMUM`), only one instance of the maximum value is listed, regardless of the number of instances of that maximum value in the set. This enum is used to pass the aggregate function argument to Component.createColumn(options), Component.createCondition(options), Query.createColumn(options), and Query.createCondition(options). **Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Type	enum
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.1

Values

Value	Description
AVERAGE	Calculates the average value.
AVERAGE_DISTINCT	Calculates the average distinct value.
COUNT	Counts the number of results.
COUNT_DISTINCT	Counts the number of distinct results.
MAXIMUM	Determines the maximum value. If the values are dates, the most recent date is determined.
MAXIMUM_DISTINCT	Determines the maximum distinct value. If the values are dates, the most recent date is determined.
MEDIAN	Calculates the median value.
MINIMUM	Determines the minimum value. If the values are dates, the earliest date is determined.

Value	Description
MINIMUM_DISTINCT	Determines the minimum distinct value. If the values are dates, the earliest date is determined.
SUM	Adds all values.
SUM_DISTINCT	Adds all distinct values.

query.Operator

Enum Description	Holds the string values for operators supported with the N/query Module. This enum is used to pass the operator argument to Query.createCondition(options) and Component.createCondition(options). **Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Type	enum
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.1

Values

Value
AFTER
AFTER_NOT
ANY_OF
ANY_OF_NOT
BEFORE
BEFORE_NOT
BETWEEN
BETWEEN_NOT
CONTAIN
CONTAIN_NOT
EMPTY
EMPTY_NOT
ENDWITH
ENDWITH_NOT
EQUAL

Value
EQUAL_NOT
GREATER
GREATER_NOT
GREATER_OR_EQUAL
GREATER_OR_EQUAL_NOT
IS
IS_NOT
LESS
LESS_NOT
LESS_OR_EQUAL
LESS_OR_EQUAL_NOT
ON
ON_NOT
ON_OR_AFTER
ON_OR_AFTER_NOT
ON_OR_BEFORE
ON_OR_BEFORE_NOT
START_WITH
START_WITH_NOT
WITHIN
WITHIN_NOT

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

query.ReturnType

Enum Description	Holds the string values for the formula return types supported with the N/query Module. This enum is used to pass the formula return type argument to Query.createColumn(options), Component.createColumn(options), Query.createCondition(options), and Component.createCondition(options). For more information on formulas, see the help topics SuiteAnalytics Workbook Beta, SQL Expressions, and Search Formula Examples and Tips. **Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Type	enum
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.1

Values

Value
ANY
BOOLEAN
CURRENCY
DATE
DATETIME
DURATION

Value
FLOAT
INTEGER
KEY
RELATIONSHIP
STRING
UNKNOWN

query.SortLocale

Enum Description	Holds the string values for sort locales supported with the N/query Module. This enum is used to pass the locale argument to Query.createSort(options) and Component.createSort(options). **Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Type	enum
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.2

Values

Value
ARABIC
ARABIC_ABJ_MATCH
ARABIC_ABJ_MATCH_CI
ARABIC_ABJ_SORT
ARABIC_ABJ_SORT_CI
ARABIC_CI
ARABIC_MATCH
ARABIC_MATCH_CI
ASCII7
ASCII7_CI
AZERBAIJANI
AZERBAIJANI_CI
BENGALI

Value
BENGALI_CI
BIG5
BIG5_CI
BINARY
BINARY_CI
BULGARIAN
BULGARIAN_CI
CANADIAN_M
CATALAN
CATALAN_CI
CROATIAN
CROATIAN_CI
CS_CZ
CZECH
CZECH_CI
CZECH_PUNCTUATION
CZECH_PUNCTUATION_CI
DANISH
DANISH_CI
DANISH_M
DA_DK
DE_DE
DUTCH
DUTCH_CI
EBCDIC
EBCDIC_CI
EEC_EURO
EEC_EUROPA3
EEC_EUROPA3_CI
EEC_EURO_CI
EN
EN_AU
EN_CA
EN_GB

Value
EN_US
ESTONIAN
ESTONIAN_CI
ES_AR
ES_ES
FINNISH
FINNISH_CI
FRENCH
FRENCH_AI
FRENCH_CI
FRENCH_M
FR_CA
FR_FR
GBK
GBK_AI
GBK_CI
GENERIC_M
GERMAN
GERMAN_AI
GERMAN_CI
GERMAN_DIN
GERMAN_DIN_AI
GERMAN_DIN_CI
GREEK
GREEK_AI
GREEK_CI
HEBREW
HEBREW_AI
HEBREW_CI
HE_IL
HKSCS
HKSCS_AI
HKSCS_CI
HUNGARIAN

Value
HUNGARIAN_AI
HUNGARIAN_CI
ICELANDIC
ICELANDIC_AI
ICELANDIC_CI
INDONESIAN
INDONESIAN_AI
INDONESIAN_CI
ITALIAN
ITALIAN_AI
ITALIAN_CI
IT_IT
JAPANESE_M
JA_JP
KOREAN_M
KO_KR
LATIN
LATIN_AI
LATIN_CI
LATVIAN
LATVIAN_AI
LATVIAN_CI
LITHUANIAN
LITHUANIAN_AI
LITHUANIAN_CI
MALAY
MALAY_AI
MALAY_CI
NL_NL
NORWEGIAN
NORWEGIAN_AI
NORWEGIAN_CI
POLISH
POLISH_AI

Value
POLISH_CI
PT_BR
PUNCTUATION
PUNCTUATION_AI
PUNCTUATION_CI
ROMANIAN
ROMANIAN_AI
ROMANIAN_CI
RUSSIAN
RUSSIAN_AI
RUSSIAN_CI
RU_RU
SCHINESE_PINYIN_M
SCHINESE_RADICAL_M
SCHINESE_STROKE_M
SLOVAK
SLOVAK_AI
SLOVAK_CI
SLOVENIAN
SLOVENIAN_AI
SLOVENIAN_CI
SPANISH
SPANISH_AI
SPANISH_CI
SPANISH_M
SV_SE
SWEDISH
SWEDISH_AI
SWEDISH_CI
SWISS
SWISS_AI
SWISS_CI
TCHINESE_RADICAL_M
TCHINESE_STROKE_M

Value
THAI_M
TH_TH
TR_TR
TURKISH
TURKISH_AI
TURKISH_CI
UKRAINIAN
UKRAINIAN_AI
UKRAINIAN_CI
UNICODE_BINARY
UNICODE_BINARY_AI
UNICODE_BINARY_CI
VIETNAMESE
VIETNAMESE_AI
VIETNAMESE_CI
WEST_EUROPEAN
WEST_EUROPEAN_AI
WEST_EUROPEAN_CI
ZH_CN
ZH_TW

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

query.Type

Important: For the 2018.2 release, the N/query module supports the same record types supported by the SuiteAnalytics Workbook UI. For more information, see the help topics SuiteAnalytics Workbook Beta and Supported Record Types for the SuiteAnalytics Workbook Beta Period.

Enum Description	Holds the string values for search types used in the query definition. This enum is used to pass the initial search type argument to query.create(options). **Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Type	enum
Module	N/query Module
Sibling Module Members	N/query Module Members
Since	2018.1

Values

Note: Before using these values, consider the following:

- A search type is not the same as a record type. The supported search types listed below do not necessarily correspond with the supported record types listed in the N/record Module.
- Depending on your account and role, some of these values might not be available.

Enum Value	Sets Query.type Property To
ACCOUNT	account
ACCOUNTING_CONTEXT	accountingcontext
ACCOUNTING_PERIOD	accountingperiod
ADVANCED_REV_REC_PLUGIN	advancedrevrecplugin
ADV_INTERCOMPANY_JOURNAL_ENTRY	advintercompanyjournalentry
ALLOCATION_METHOD	allocationmethod
AMORTIZATION_SCHEDULE	amortizationschedule
AMORTIZATION_TEMPLATE	amortizationtemplate
ANOTHER_HIERARCHY_RECORD	anotherhierarchyrecord

Enum Value	Sets Query.type Property To
BANK_CONNECTIVITY_PLUGIN	bankconnectivityplugin
BILLING_CLASS	billingclass
BILLING_SCHEDULE	billingschedule
BRANCHRECORD	branchrecord
BUDGETCATEGORY	budgetcategory
BUDGETEXCHANGERATE	budgetexchangerate
BUDGETIMPORT	budgetimport
BUDGETS	budgets
BULK_PROC_SUBMISSION	bulkprocsubmission
BUNDLE_INSTALLATION_SCRIPT	bundleinstallationscript
BUNDLE_INSTALLATION_SCRIPT_DEPLOYMENT	bundleinstallationscriptdeployment
BUYING_REASON	buyingreason
BUYING_TIME_FRAME	buyingtimeframe
CALENDAR_EVENT	calendarevent
CAMPAIGN_AUDIENCE	campaignaudience
CAMPAIGN_CATEGORY	campaigncategory
CAMPAIGN_CHANNEL	campaignchannel
CAMPAIGN_EMAIL_ADDRESS	campaignemailaddress
CAMPAIGN_EVENT	campaignevent
CAMPAIGN_FAMILY	campaignfamily
CAMPAIGN_OFFER	campaignoffer
CAMPAIGN_RESPONSE	campaignresponse
CAMPAIGN_SEARCH_ENGINE	campaignsearchengine
CAMPAIGN_TEMPLATE	campaigntemplate
CAMPAIGN_VERTICAL	campaignvertical
CASE_PROFILE	caseprofile
CASH_REFUND	cashrefund
CASH_SALE	cashsale
CATEGORY1099MISC	category1099misc
CHECK	check
CLASSIFICATION	classification
CLIENT_SCRIPT	clientscript
CLIENT_SCRIPT_DEPLOYMENT	clientscriptdeployment
CLOB_RECORD	clobrecord

Enum Value	Sets Query.type Property To
COMPANY	company
COMPETITOR	competitor
COMPOSITE_KEY_SOURCE_RECORD	compositekeysourcerecord
COMPOSITE_RECORD	compositerecord
CONSOLIDATEDEXCHANGERATE	consolidatedexchangerate
CONSOLIDATEDEXCHANGERATEINTERNAL	consolidatedexchangerateinternal
CONSOLIDATED_RATE_ADJUSTOR_PLUGIN	consolidatedrateadjustorplugin
CONSOLIDATION_ACCOUNT	consolidationaccount
CONSOLIDATION_ACCOUNT_TYPE	consolidationaccounttype
CONSOLIDATION_BUDGET_RATE	consolidationbudgetrate
CONSOLIDATION_CURRENCY	consolidationcurrency
CONSOLIDATION_RATE	consolidationrate
CONSOLIDATION_SUBSIDIARY	consolidationsubsidiary
CONSOLIDATION_TRANSACTION	consolidationtransaction
CONSUMER_SPECIFIC_RECORD_TYPE	consumerspecificrecordtype
CONTACT	contact
CONTACT_CATEGORY	contactcategory
CONTACT_ROLE	contactrole
COUPON_CODE	couponcode
COURSE_RECORD	courserecord
CREDIT_CARDS	creditcards
CREDIT_CARD_CHARGE	creditcardcharge
CREDIT_CARD_REFUND	creditcardrefund
CREDIT_MEMO	creditmemo
CRM_TEMPLATE	crmtemplate
CRM_TEMPLATE_CATEGORY	crmtemplatecategory
CURRENCY	currency
CURRENCY_FIELD_RECORD	currencyfieldrecord
CURRENCY_FIELD_TYPE	currencyfieldtype
CURRENCY_RATE	currencyrate
CUSTOM	custom
CUSTOMER	customer
CUSTOMER_CATEGORY	customercategory
CUSTOMER_CHARGE	customercharge

Enum Value	Sets Query.type Property To
CUSTOMER_DEPOSIT	customerdeposit
CUSTOMER_MESSAGE	customermessage
CUSTOMER_PAYMENT	customerpayment
CUSTOMER_REFUND	customerrefund
CUSTOMER_STATUS	customerstatus
CUSTOMRECORD1	customrecord1
CUSTOM_GL_PLUGIN	customglplugin
CUSTOM_LIST	customlist
CUSTOM_RECORD_TYPE	customrecordtype
DATE_FIELD_TYPE	datefieldtype
DATE_RECORD	daterecord
DATE_TIME_RECORD	datetimerecord
DATE_TIME_ZONE	datetimezone
DEFAULTING_PORTED_RECORD	defaultingportedrecord
DEF_VIEW_TEST_RECORD	defviewtestrecord
DELETED_RECORD	deletedrecord
DEPARTMENT	department
DEPOSIT	deposit
DEPOSIT_APPLICATION	depositapplication
DESCRIPTION_ITEM	descriptionitem
DEVICE_ID	deviceid
DISABLEDCHANNELFORMTESTRECORD	disabledchannelformtestrecord
DISCOUNT_ITEM	discountitem
DISPLAY_INACTIVE_TEST_RECORD	displayinactivetestrecord
DOMAIN	domain
DOWNLOAD_ITEM	downloaditem
DURATION_RECORD	durationrecord
EMAIL_CAPTURE_PLUGIN	emailcaptureplugin
EMAIL_TEMPLATE	emailtemplate
EMPLOYEE	employee
EMPLOYEE_LIST	employeelist
EMPLOYEE_STATUS	employeestatus
END_TO_END_TIME	endtoendtime
ENTITY	entity

Enum Value	Sets Query.type Property To
ENTITY_GROUP	entitygroup
ESCALATION_TERRITORY	escalationterritory
ESTIMATE	estimate
EXAMPLE_TRANSACTION	exampletransaction
EXPENSE_CATEGORY	expensecategory
EXPENSE_REPORT	expensereport
EXPOSURENOTLIMITEDRECORD	exposurenotlimitedrecord
FACULTYRECORD	facultyrecord
FAX_TEMPLATE	faxtemplate
FIELD_LABEL	fieldlabel
FILE	file
FLOAT_NUMBERS_TEST_RECORD	floatnumberstestrecord
FORECAST	forecast
FORMULA_POLYMORPHIC_RECORD	formulapolymorphicrecord
FORMULA_RECORD	formularecord
FULFILLMENT_EXCEPTION_REASON	fulfillmentexceptionreason
FX_REVAL	fxreval
GATEWAY_NOTIFICATION	gatewaynotification
GENERAL_ALLOCATION_SCHEDULE	generalallocationschedule
GENERIC_RESOURCE	genericresource
GENERIC_TEST_RECORD	generictestrecord
GIFT_CERTIFICATE	giftcertificate
GIFT_CERTIFICATE_ITEM	giftcertificateitem
HIERARCHY_RECORD	hierarchyrecord
HYBRID_RECORD_LOG	hybridrecordlog
INCOTERM	incoterm
INTEGRATION_APP	integrationapp
INTERNAL_ID_TEST_RECORD	internalidtestrecord
INVENTORY_ADJUSTMENT	inventoryadjustment
INVENTORY_DISTRIBUTION	inventorydistribution
INVENTORY_ITEM	inventoryitem
INVENTORY_TRANSFER	inventorytransfer
INVENTORY_WORKSHEET	inventoryworksheet
INVOICE	invoice

Enum Value	Sets Query.type Property To
INVT_ITEM_PRICE_HISTORY	invtitempricehistory
ISSUE	issue
ISSUE_EXTERNAL_STATUS	issueexternalstatus
ISSUE_PRIORITY	issuepriority
ISSUE_PRODUCT	issueproduct
ISSUE_REPRODUCIBILITY	issuereproducibility
ISSUE_ROLE	issuerole
ISSUE_SEVERITY	issueseverity
ISSUE_SOURCE	issuesource
ISSUE_STATUS	issuestatus
ISSUE_TAG	issuetag
ISSUE_TRACK_CODE	issuetrackcode
ISSUE_TYPE	issuetype
ITEM	item
ITEM_FULFILLMENT	itemfulfillment
ITEM_GROUP	itemgroup
ITEM_RECEIPT	itemreceipt
I_P_RESTRICTIONS	iprestrictions
JOB	job
JOB_RESOURCE_ROLE	jobresourcerole
JOB_STATUS	jobstatus
JOB_TYPE	jobtype
JOURNAL	journal
KIT_ITEM	kititem
KNOWLEDGE_BASE	knowledgebase
LOCATION	location
LOCATION_COSTING_GROUP	locationcostinggroup
LOGIN_AUDIT	loginaudit
MAIL_TEMPLATE	mailtemplate
MAP_REDUCE_SCRIPT	mapreducescript
MAP_REDUCE_SCRIPT_DEPLOYMENT	mapreducescriptdeployment
MARKUP_ITEM	markupitem
MASS_UPDATE_SCRIPT	massupdatescript
MASS_UPDATE_SCRIPT_DEPLOYMENT	massupdatescriptdeployment

Enum Value	Sets Query.type Property To
MATERIALIZED_HIERARCHY_RECORD	materializedhierarchyrecord
MEDIA_ITEM_FOLDER	mediaitemfolder
MEM_DOC	memdoc
MEM_DOC_TRANSACTION_TEMPLATE	memdoctransactiontemplate
MESSAGE	message
NAMED_GROUP_RECORD	namedgrouprecord
NEXUS	nexus
NON_INVENTORY_PURCHASE_ITEM	noninventorypurchaseitem
NON_INVENTORY_RESALE_ITEM	noninventoryresaleitem
NON_INVENTORY_SALE_ITEM	noninventorysaleitem
NOTE	note
NOTE_TYPE	notetype
NUMERIC_RECORD	numericrecord
ONLINE_CASE_FORM	onlinecaseform
ONLINE_FORM_TEMPLATE	onlineformtemplate
ONLINE_LEAD_FORM	onlineleadform
OPPORTUNITY	opportunity
OTHER_CHARGE_PURCHASE_ITEM	otherchargepurchaseitem
OTHER_CHARGE_RESALE_ITEM	otherchargeresaleitem
OTHER_CHARGE_SALE_ITEM	otherchargesaleitem
OTHER_NAME	othername
OTHER_NAME_CATEGORY	othernamecategory
PAGE	page
PAGINATION_RECORD	paginationrecord
PARTNER	partner
PARTNER_CATEGORY	partnercategory
PAYCHECK	paycheck
PAYMENT_EVENT	paymentevent
PAYMENT_GATEWAY_PLUGIN	paymentgatewayplugin
PAYMENT_ITEM	paymentitem
PAYMENT_METHOD	paymentmethod
PAYMENT_PROCESSING_PROFILE	paymentprocessingprofile
PAYROLL_ITEM	payrollitem
PDF_TEMPLATE	pdftemplate

Enum Value	Sets Query.type Property To
PERSISTED_RECORD	persistedrecord
PERSISTED_RECORD_FULL_JOIN	persistedrecordfulljoin
PERSISTED_RECORD_INVALID_TABLE	persistedrecordinvalidtable
PERSISTED_RECORD_NO_CREATE	persistedrecordnocreate
PERSISTED_RECORD_NO_DELETE	persistedrecordnodelete
PERSISTED_RECORD_NO_EDIT	persistedrecordnoedit
PERSISTED_RECORD_NO_LOAD	persistedrecordnoload
PERSISTED_RECORD_RIGHT_JOIN	persistedrecordrightjoin
PERSISTED_RECORD_SIMPLE_OPTIONS	persistedrecordsimpleoptions
PERSISTED_RECORD_U_Q_KEY_REF	persistedrecorduqkeyref
PERSISTED_RECORD_U_Q_KEY_REF_TYPE	persistedrecorduqkeyreftype
PHONE_CALL	phonecall
PLUG_IN_TYPE	plugintype
PLUG_IN_TYPE_IMPL	plugintypeimpl
PORTLET	portlet
PORTLET_DEPLOYMENT	portletdeployment
PRICE_LEVEL	pricelevel
PRICING	pricing
PRICING_GROUP	pricinggroup
PRIMARY_RECORD	primaryrecord
PROJECT_TASK	projecttask
PROJECT_TEMPLATE	projecttemplate
PROMOTIONS_PLUGIN	promotionsplugin
PROMOTION_CODE	promotioncode
PUBLISHED_SAVED_SEARCH	publishedsavedsearch
PURCHASE_ORDER	purchaseorder
PURCHASE_REQUISITION	purchaserequisition
QUANTITY_PRICING_SCHEDULE	quantitypricingschedule
QUOTA	quota
RECENT_RECORD	recentrecord
RECORD_SERVICE_TEST_RECORD	recordservicetestrecord
RECORD_TYPE	recordtype
RECORD_WITH_HIERARCHY_RELATIONSHIP	recordwithhierarchyrelationship
REDIRECT	redirect

Enum Value	Sets Query.type Property To
REGION	region
RELATIONSHIP_DISPLAY_INACTIVE	relationshipdisplayinactive
RELATIONSHIP_SELECT_EMPLOYEE_RECORD	relationshipselectemployeerecord
REPORT_DEFINITION	reportdefinition
REQUEST_LEVEL_RECORD1	requestlevelrecord1
REQUEST_LEVEL_RECORD2	requestlevelrecord2
RESOURCE	resource
RESTLET	restlet
RESTLET_DEPLOYMENT	restletdeployment
RESTRICTIONS_ONCE_REMOVED	restrictionsonceremoved
RESTRICTIONS_TWICE_REMOVED	restrictionstwiceremoved
RESTRICTION_ANNOTATION_TEST_RECORD	restrictionannotationtestrecord
RESTRICTION_TEST_RECORD	restrictiontestrecord
RETURN_AUTHORIZATION	returnauthorization
REV_REC_SCHEDULE	revrecschedule
REV_REC_TEMPLATE	revrectemplate
ROLE	role
RSTR_ALT_LOCATION	rstraltlocation
RSTR_LOCATION	rstrlocation
RSTR_RECORD	rstrrecord
SALES	sales
SALES_ORDER	salesorder
SALES_READINESS	salesreadiness
SALES_ROLE	salesrole
SALES_TAX_ITEM	salestaxitem
SALES_TERRITORY	salesterritory
SALES_TRANSACTION	salestransaction
SAMPLE_RECORD	samplerecord
SCHEDULED_SCRIPT	scheduledscript
SCHEDULED_SCRIPT_DEPLOYMENT	scheduledscriptdeployment
SCHEDULED_SCRIPT_INSTANCE	scheduledscriptinstance
SCRIPT	script
SCRIPTING_TEST_RECORD	scriptingtestrecord
SCRIPTING_TEST_RECORD_SUBRECORD2_TARGET	scriptingtestrecordsubrecord2target

Enum Value	Sets Query.type Property To
SCRIPTING_TEST_RECORD_SUBRECORD2_TARGET2	scriptingtestrecordsubrecord2target2
SCRIPTING_TEST_RECORD_SUBRECORD3_TARGET	scriptingtestrecordsubrecord3target
SCRIPTING_TEST_RECORD_SUBRECORD3_TARGET2	scriptingtestrecordsubrecord3target2
SCRIPTING_TEST_RECORD_SUBRECORD4_TARGET	scriptingtestrecordsubrecord4target
SCRIPTING_TEST_RECORD_SUBRECORD4_TARGET2	scriptingtestrecordsubrecord4target2
SCRIPTING_TEST_RECORD_SUBRECORD_TARGET	scriptingtestrecordsubrecordtarget
SCRIPTING_TEST_RECORD_SUBRECORD_TARGET2	scriptingtestrecordsubrecordtarget2
SCRIPTING_TEST_RECORD_TARGET	scriptingtestrecordtarget
SCRIPTING_TEST_RECORD_TARGET2	scriptingtestrecordtarget2
SCRIPT_DEPLOYMENT	scriptdeployment
SCRIPT_NOTE	scriptnote
SCRIPT_RECORD_TYPE	scriptrecordtype
SCRIP_INH_TEST_RECORD1	scripinhtestrecord1
SCRIP_INH_TEST_RECORD2	scripinhtestrecord2
SCRIP_INH_TEST_RECORD3	scripinhtestrecord3
SCRIP_INH_TEST_RECORD4	scripinhtestrecord4
SEARCH_CAMPAIGN	searchcampaign
SEARCH_SCHEDULE	searchschedule
SEARCH_URL_TEST_SOURCE_RECORD	searchurltestsourcerecord
SEARCH_URL_TEST_TARGET_RECORD	searchurltesttargetrecord
SELECT_OPTIONS_SOURCE_RECORD	selectoptionssourcerecord
SERVICE_PURCHASE_ITEM	servicepurchaseitem
SERVICE_RESALE_ITEM	serviceresaleitem
SERVICE_SALE_ITEM	servicesaleitem
SHIPPING_PACKAGE	shippingpackage
SHIPPING_PARTNERS_PLUGIN	shippingpartnersplugin
SHIP_ITEM	shipitem
SHOPPING_CART	shoppingcart
SIMPLE_RECORD	simplerecord
SIMPLE_RECORD_LOCATION	simplerecordlocation
SITE_CATEGORY	sitecategory
SLAVE	slave
SLAVE_EMPTY_EXPRESSION	slaveemptyexpression
SLAVE_FEATURE	slavefeature

Enum Value	Sets Query.type Property To
SLAVE_MASTER_PERMISSION	slavemasterpermission
SLAVE_PERMISSION	slavepermission
SLAVE_TARGET_PROPERTY	slavetargetproperty
SLAVE_VALID_EXPRESSION	slavevalidexpression
SOLUTION	solution
SORT_BASE_RECORD	sortbaserecord
SORT_RECORD	sortrecord
SORT_RELATED_RECORD	sortrelatedrecord
STATIC_LIST_RECORD	staticlistrecord
STATIC_OPTIONS_FIELD_TEST_RECORD	staticoptionsfieldtestrecord
STATIC_OPTIONS_VALUE	staticoptionsvalue
STORE_TAB	storetab
STUDENT_RECORD	studentrecord
SUBLIST	sublist
SUBSIDIARY	subsidiary
SUBTOTAL_ITEM	subtotalitem
SUB_SELECT_GROUP_RECORD	subselectgrouprecord
SUITELET	suitelet
SUITELET_DEPLOYMENT	suiteletdeployment
SUITE_SCRIPT_DETAIL	suitescriptdetail
SUPPORT_CASE	supportcase
SUPPORT_CASE_ISSUE	supportcaseissue
SUPPORT_CASE_ORIGIN	supportcaseorigin
SUPPORT_CASE_PRIORITY	supportcasepriority
SUPPORT_CASE_STATUS	supportcasestatus
SUPPORT_CASE_TYPE	supportcasetype
SUPPORT_TERRITORY	supportterritory
SYSTEM_EMAIL_TEMPLATE	systememailtemplate
SYSTEM_JOURNAL	systemjournal
SYSTEM_NOTE	systemnote
SYSTEM_NOTE_FIELD	systemnotefield
TABLE_CONDITION_TEST_RECORD	tableconditiontestrecord
TASK	task
TASK_ITEM_STATUS	taskitemstatus

Enum Value	Sets Query.type Property To
TAX_CALCULATION_PLUGIN	taxcalculationplugin
TAX_ITEM_TAX_GROUP	taxitemtaxgroup
TAX_PERIOD	taxperiod
TAX_TYPE	taxtype
TERM	term
TESTDOAGGREGATEDOSUBTYPE	testdoaggregatedosubtype
TESTDOAGGREGATERESTRICTIONRECORD	testdoaggregaterestrictionrecord
TEST_COMPOSED_RECORD1	testcomposedrecord1
TEST_COMPOSED_RECORD2	testcomposedrecord2
TEST_COMPOSED_RECORD3	testcomposedrecord3
TEST_CONFIGURABLE_RECORD	testconfigurablerecord
TEST_DO_AGGREGATE_RECORD	testdoaggregaterecord
TEST_EXPOSURE_RECORD	testexposurerecord
TEST_FEATURE_RECORD	testfeaturerecord
TEST_FULL_RECORD	testfullrecord
TEST_MACROS_UMD_RECORD	testmacrosumdrecord
TEST_MULTI_TABLE_PERSISTENCE_RECORD	testmultitablepersistencerecord
TEST_NEW_URLS_RECORD	testnewurlsrecord
TEST_NEW_URLS_TARGET_RECORD	testnewurlstargetrecord
TEST_NEW_URLS_UNSUPPORTED_RECORD	testnewurlsunsupportedrecord
TEST_NEXT_STANDARD_RECORD	testnextstandardrecord
TEST_PLUGIN	testplugin
TEST_PRIMARY_RECORD_FOR_RELATIONSHIPS	testprimaryrecordforrelationships
TEST_RECORD	testrecord
TEST_RECORD_ACTION_RECORD	testrecordactionrecord
TEST_RECORD_UNIQUE_KEY	testrecorduniquekey
TEST_RECORD_WITHOUT_LABEL	testrecordwithoutlabel
TEST_RECORD_WITH_DISABLED_RECORD_SORT_FIELDS	testrecordwithdisabledrecordsortfields
TEST_RECORD_WITH_SORT_FIELDS	testrecordwithsortfields
TEST_REGRESSION_RECORD	testregressionrecord
TEST_RELATED_PROPERTY	testrelatedproperty
TEST_SECURED_RECORD	testsecuredrecord
TEST_SIMPLE_PERSISTENCE_RECORD	testsimplepersistencerecord
TEST_SIMPLE_PERSISTENCE_SELECT_SIDE_RECORD	testsimplepersistenceselectsiderecord

Enum Value	Sets Query.type Property To
TEST_SLAVING_RATE_FIELD_RECORD	testslavingratefieldrecord
TEST_SLAVING_RECORD	testslavingrecord
TEST_SLAVING_RECORD_OPTIMIZED	testslavingrecordoptimized
TEST_SOURCING_MASTER_FIELD_ANNOTATION_MASTER	testsourcingmasterfieldannotationmaster
TEST_SOURCING_MASTER_FIELD_ANNOTATION_RECORD	testsourcingmasterfieldannotationrecord
TEST_SOURCING_OPTIONS_CONDITION_MASTER	testsourcingoptionsconditionmaster
TEST_SOURCING_OPTIONS_CONDITION_RECORD	testsourcingoptionsconditionrecord
TEST_SOURCING_OPTIONS_CONDITION_TARGET	testsourcingoptionsconditiontarget
TEST_SOURCING_SUBLIST_FIELD_ANNOTATION_MASTER	testsourcingsublistfieldannotationmaster
TEST_SOURCING_SUBLIST_FIELD_ANNOTATION_RECORD	testsourcingsublistfieldannotationrecord
TEST_SOURCING_VALUE_RATE_COL_MASTER	testsourcingvalueratecolmaster
TEST_SOURCING_VALUE_RATE_COL_RECORD	testsourcingvalueratecolrecord
TEST_STANDARD_RECORD	teststandardrecord
TEST_TRANSACTION	testtransaction
TIME_BILL	timebill
TOPIC	topic
TRACKING_NUMBER	trackingnumber
TRANSACTION	transaction
TRANSACTION_ADDRESSBOOK	transactionaddressbook
TRANSACTION_BILLING_ADDRESSBOOK	transactionbillingaddressbook
TRANSACTION_NUMBERING_AUDIT_LOG	transactionnumberingauditlog
TRANSACTION_RETURN_ADDRESSBOOK	transactionreturnaddressbook
TRANSACTION_SHIPPING_ADDRESSBOOK	transactionshippingaddressbook
TRANSFER	transfer
TRANSFER_ORDER	transferorder
TWO_FACTOR_DEVICE	twofactordevice
TYPE_FIELD_PARENT_RECORD	typefieldparentrecord
TYPE_FIELD_RECORD	typefieldrecord
UMD_FIELD	umdfield
UNDELIVERED_EMAIL	undeliveredemail
UNIFICATION_TEST	unificationtest
USER_EVENT_SCRIPT	usereventscript
USER_EVENT_SCRIPT_DEPLOYMENT	usereventscriptdeployment
USRCATEGORY	usrcategory

Enum Value	Sets Query.type Property To
USRSAVEDSEARCH	usrsavedsearch
USR_ANALYTICAL	usranalytical
USR_AUDITING_SOURCE_RECORD	usrauditingsourcerecord
USR_AUDIT_LOG	usrauditlog
USR_CHANNEL_AG_BTH_ROOT	usrchannelagbthroot
USR_CHANNEL_AG_BTH_ROOT_SUB_TYPE	usrchannelagbthrootsubtype
USR_CHANNEL_AG_BTH_SEARCH_MTM_ROOT	usrchannelagbthsearchmtmroot
USR_CHANNEL_AG_BTH_SEARCH_MTM_SUB_TYPE	usrchannelagbthsearchmtmsubtype
USR_CHANNEL_AG_BTH_SEARCH_MTO_ROOT	usrchannelagbthsearchmtoroot
USR_CHANNEL_AG_BTH_SEARCH_MTO_SUB_TYPE	usrchannelagbthsearchmtosubtype
USR_CHANNEL_AG_SRC_ROOT	usrchannelagsrcroot
USR_CHANNEL_AG_SRC_ROOT_SUB_TYPE	usrchannelagsrcrootsubtype
USR_CHANNEL_AG_SRC_SEARCH_MTM_PRIMARY	usrchannelagsrcsearchmtmprimary
USR_CHANNEL_AG_SRC_SEARCH_MTO_PRIMARY	usrchannelagsrcsearchmtoprimary
USR_CHANNEL_AG_TGT_ROOT	usrchannelagtgtroot
USR_CHANNEL_AG_TGT_SEARCH_MTM_ROOT	usrchannelagtgtsearchmtmroot
USR_CHANNEL_AG_TGT_SEARCH_MTM_SUB_TYPE	usrchannelagtgtsearchmtmsubtype
USR_CHANNEL_AG_TGT_SEARCH_MTO_ROOT	usrchannelagtgtsearchmtoroot
USR_CHANNEL_AG_TGT_SEARCH_MTO_SUB_TYPE	usrchannelagtgtsearchmtosubtype
USR_CHANNEL_STD_ROOT	usrchannelstdroot
USR_CHANNEL_STD_SEARCH_MTM_PRIMARY	usrchannelstdsearchmtmprimary
USR_CHANNEL_STD_SEARCH_MTO_PRIMARY	usrchannelstdsearchmtoprimary
USR_EXECUTION_LOG	usrexecutionlog
USR_EXPOSE_EXTERNAL	usrexposeexternal
USR_EXPOSE_IMPORTANT	usrexposeimportant
USR_EXPOSE_INTNL_FLD_PLAIN_AG_TGT_PLAIN_MTO_ROOT	usrexposeintnlfldplainagtgtplainmtoroot
USR_EXPOSE_INTNL_FLD_PLAIN_AG_TGT_PLAIN_MTO_SUB_TYPE	usrexposeintnlfldplainagtgtplainmtosubtype
USR_EXPOSE_INTNL_FLD_PLAIN_AG_TGT_ROOT	usrexposeintnlfldplainagtgtroot
USR_EXPOSE_INTNL_FLD_PLAIN_STD_N_VAL_MTO_PRIMARY	usrexposeintnlfldplainstdnvalmtoprimary
USR_EXPOSE_INTNL_FLD_PLAIN_STD_ROOT	usrexposeintnlfldplainstdroot
USR_EXPOSE_PLAIN_FLD_INTNL_AG_BTH_N_VAL_MTO_ROOT	usrexposeplainfldintnlagbthnvalmtoroot
USR_EXPOSE_PLAIN_FLD_INTNL_AG_BTH_N_VAL_MTO_SUB_TYPE	usrexposeplainfldintnlagbthnvalmtosubtype

Enum Value	Sets Query.type Property To
USR_EXPOSE_PLAIN_FLD_INTNL_AG_BTH_PLAIN_MTO_ROOT	usrexposeplainfldintnlagbthplainmtoroot
USR_EXPOSE_PLAIN_FLD_INTNL_AG_BTH_PLAIN_MTO_SUB_TYPE	usrexposeplainfldintnlagbthplainmtosubtype
USR_EXPOSE_PLAIN_FLD_INTNL_AG_SRC_N_VAL_MTO_PRIMARY	usrexposeplainfldintnlagsrcnvalmtoprimary
USR_EXPOSE_PLAIN_FLD_INTNL_AG_SRC_PLAIN_MTO_PRIMARY	usrexposeplainfldintnlagsrcplainmtoprimary
USR_EXPOSE_PLAIN_FLD_INTNL_AG_TGT_N_VAL_MTO_ROOT	usrexposeplainfldintnlagtgtnvalmtoroot
USR_EXPOSE_PLAIN_FLD_INTNL_AG_TGT_N_VAL_MTO_SUB_TYPE	usrexposeplainfldintnlagtgtnvalmtosubtype
USR_EXPOSE_PLAIN_FLD_INTNL_AG_TGT_PLAIN_MTO_ROOT	usrexposeplainfldintnlagtgtplainmtoroot
USR_EXPOSE_PLAIN_FLD_INTNL_AG_TGT_PLAIN_MTO_SUB_TYPE	usrexposeplainfldintnlagtgtplainmtosubtype
USR_EXPOSE_PLAIN_FLD_INTNL_STD_N_VAL_MTM_PRIMARY	usrexposeplainfldintnlstdnvalmtmprimary
USR_EXPOSE_PLAIN_FLD_INTNL_STD_N_VAL_MTO_PRIMARY	usrexposeplainfldintnlstdnvalmtoprimary
USR_EXPOSE_PLAIN_FLD_INTNL_STD_PLAIN_MTM_PRIMARY	usrexposeplainfldintnlstdplainmtmprimary
USR_EXPOSE_PLAIN_FLD_INTNL_STD_PLAIN_MTO_PRIMARY	usrexposeplainfldintnlstdplainmtoprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_BTH_N_VAL_MTO_ROOT	usrexposeplainfldplainagbthnvalmtoroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_BTH_N_VAL_MTO_SUB_TYPE	usrexposeplainfldplainagbthnvalmtosubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_BTH_PLAIN_MTO_ROOT	usrexposeplainfldplainagbthplainmtoroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_BTH_PLAIN_MTO_SUB_TYPE	usrexposeplainfldplainagbthplainmtosubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_BTH_ROOT	usrexposeplainfldplainagbthroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_BTH_SUB_TYPE	usrexposeplainfldplainagbthsubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_SRC_N_VAL_MTM_PRIMARY	usrexposeplainfldplainagsrcnvalmtmprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_SRC_N_VAL_MTO_PRIMARY	usrexposeplainfldplainagsrcnvalmtoprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_SRC_PLAIN_MTM_PRIMARY	usrexposeplainfldplainagsrcplainmtmprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_SRC_PLAIN_MTO_PRIMARY	usrexposeplainfldplainagsrcplainmtoprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_SRC_ROOT	usrexposeplainfldplainagsrcroot

Enum Value	Sets Query.type Property To
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_SRC_SUB_TYPE	usrexposeplainfldplainagsrcsubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_N_VAL_MTM_ROOT	usrexposeplainfldplainagtgtnvalmtmroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_N_VAL_MTM_SUB_TYPE	usrexposeplainfldplainagtgtnvalmtmsubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_N_VAL_MTO_ROOT	usrexposeplainfldplainagtgtnvalmtoroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_N_VAL_MTO_SUB_TYPE	usrexposeplainfldplainagtgtnvalmtosubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_PLAIN_MTM_ROOT	usrexposeplainfldplainagtgtplainmtmroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_PLAIN_MTM_SUB_TYPE	usrexposeplainfldplainagtgtplainmtmsubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_PLAIN_MTO_ROOT	usrexposeplainfldplainagtgtplainmtoroot
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_PLAIN_MTO_SUB_TYPE	usrexposeplainfldplainagtgtplainmtosubtype
USR_EXPOSE_PLAIN_FLD_PLAIN_AG_TGT_ROOT	usrexposeplainfldplainagtgtroot
USR_EXPOSE_PLAIN_FLD_PLAIN_STD_N_VAL_MTM_PRIMARY	usrexposeplainfldplainstdnvalmtmprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_STD_N_VAL_MTO_PRIMARY	usrexposeplainfldplainstdnvalmtoprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_STD_PLAIN_MTM_PRIMARY	usrexposeplainfldplainstdplainmtmprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_STD_PLAIN_MTO_PRIMARY	usrexposeplainfldplainstdplainmtoprimary
USR_EXPOSE_PLAIN_FLD_PLAIN_STD_ROOT	usrexposeplainfldplainstdroot
USR_FEATURE_AG_BTH_ROOT	usrfeatureagbthroot
USR_FEATURE_AG_BTH_ROOT_SUB_TYPE	usrfeatureagbthrootsubtype
USR_FEATURE_AG_SRC_ROOT	usrfeatureagsrcroot
USR_FEATURE_AG_SRC_ROOT_SUB_TYPE	usrfeatureagsrcrootsubtype
USR_FEATURE_AG_TGT_ROOT	usrfeatureagtgtroot
USR_FEATURE_CSM_DEFAULT_COLUMNS_RECORD	usrfeaturecsmdefaultcolumnsrecord
USR_FEATURE_CSM_IMPORTANT_JOIN_RECORD	usrfeaturecsmimportantjoinrecord
USR_FEATURE_CSM_INHERITANCE_RECORD	usrfeaturecsminheritancerecord
USR_FEATURE_CSM_USAGE_SPECIFIC_RECORD	usrfeaturecsmusagespecificrecord
USR_FEATURE_STD_ROOT	usrfeaturestdroot
USR_NON_SYSTEM_RECORD	usrnonsystemrecord
USR_PERMISSION_AG_BTH_DENIED_MTM_ROOT	usrpermissionagbthdeniedmtmroot
USR_PERMISSION_AG_BTH_DENIED_MTM_SUB_TYPE	usrpermissionagbthdeniedmtmsubtype
USR_PERMISSION_AG_BTH_DENIED_MTO_ROOT	usrpermissionagbthdeniedmtoroot

Enum Value	Sets Query.type Property To
USR_PERMISSION_AG_BTH_DENIED_MTO_SUB_TYPE	usrpermissionagbthdeniedmtosubtype
USR_PERMISSION_AG_BTH_GRANTED_MTM_ROOT	usrpermissionagbthgrantedmtmroot
USR_PERMISSION_AG_BTH_GRANTED_MTM_SUB_TYPE	usrpermissionagbthgrantedmtmsubtype
USR_PERMISSION_AG_BTH_GRANTED_MTO_ROOT	usrpermissionagbthgrantedmtoroot
USR_PERMISSION_AG_BTH_GRANTED_MTO_SUB_TYPE	usrpermissionagbthgrantedmtosubtype
USR_PERMISSION_AG_BTH_ROOT	usrpermissionagbthroot
USR_PERMISSION_AG_BTH_ROOT_SUB_TYPE	usrpermissionagbthrootsubtype
USR_PERMISSION_AG_SRC_DENIED_MTM_PRIMARY	usrpermissionagsrcdeniedmtmprimary
USR_PERMISSION_AG_SRC_DENIED_MTO_PRIMARY	usrpermissionagsrcdeniedmtoprimary
USR_PERMISSION_AG_SRC_GRANTED_MTM_PRIMARY	usrpermissionagsrcgrantedmtmprimary
USR_PERMISSION_AG_SRC_GRANTED_MTO_PRIMARY	usrpermissionagsrcgrantedmtoprimary
USR_PERMISSION_AG_SRC_ROOT	usrpermissionagsrcroot
USR_PERMISSION_AG_SRC_ROOT_SUB_TYPE	usrpermissionagsrcrootsubtype
USR_PERMISSION_AG_TGT_DENIED_MTM_ROOT	usrpermissionagtgtdeniedmtmroot
USR_PERMISSION_AG_TGT_DENIED_MTM_SUB_TYPE	usrpermissionagtgtdeniedmtmsubtype
USR_PERMISSION_AG_TGT_DENIED_MTO_ROOT	usrpermissionagtgtdeniedmtoroot
USR_PERMISSION_AG_TGT_DENIED_MTO_SUB_TYPE	usrpermissionagtgtdeniedmtosubtype
USR_PERMISSION_AG_TGT_GRANTED_MTM_ROOT	usrpermissionagtgtgrantedmtmroot
USR_PERMISSION_AG_TGT_GRANTED_MTM_SUB_TYPE	usrpermissionagtgtgrantedmtmsubtype
USR_PERMISSION_AG_TGT_GRANTED_MTO_ROOT	usrpermissionagtgtgrantedmtoroot
USR_PERMISSION_AG_TGT_GRANTED_MTO_SUB_TYPE	usrpermissionagtgtgrantedmtosubtype
USR_PERMISSION_AG_TGT_ROOT	usrpermissionagtgtroot
USR_PERMISSION_STD_DENIED_MTM_PRIMARY	usrpermissionstddeniedmtmprimary
USR_PERMISSION_STD_DENIED_MTO_PRIMARY	usrpermissionstddeniedmtoprimary
USR_PERMISSION_STD_GRANTED_MTM_PRIMARY	usrpermissionstdgrantedmtmprimary
USR_PERMISSION_STD_GRANTED_MTO_PRIMARY	usrpermissionstdgrantedmtoprimary
USR_PERMISSION_STD_ROOT	usrpermissionstdroot
USR_POLYMORPHIC_CHILD_ONE_RECORD	usrpolymorphicchildonerecord
USR_POLYMORPHIC_CHILD_TWO_RECORD	usrpolymorphicchildtworecord
USR_POLYMORPHIC_JOIN_TEST_RECORD	usrpolymorphicjointestrecord
USR_TARGET_PROPERTIES_GROUP_BY_TARGET_RECORD	usrtargetpropertiesgroupbytargetrecord
USR_TARGET_PROPERTIES_MTO2_TARGET_RECORD	usrtargetpropertiesmto2targetrecord
USR_TARGET_PROPERTIES_MTO_TARGET_RECORD	usrtargetpropertiesmtotargetrecord
USR_TARGET_PROPERTIES_ROOT_RECORD	usrtargetpropertiesrootrecord

Enum Value	Sets Query.type Property To
USR_UNIVERSAL	usruniversal
VENDOR	vendor
VENDOR_BILL	vendorbill
VENDOR_CATEGORY	vendorcategory
VENDOR_CREDIT	vendorcredit
VENDOR_PAYMENT	vendorpayment
VENDOR_SUBSIDIARY_RELATIONSHIP	vendorsubsidiaryrelationship
WEBAPP	webapp
WEB_SITE	website
WIN_LOSS_REASON	winlossreason
WORKFLOW_ACTION_SCRIPT	workflowactionscript
WORKFLOW_ACTION_SCRIPT_DEPLOYMENT	workflowactionscriptdeployment
WORKPLACE	workplace
WORK_CALENDAR	workcalendar

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
    type: query.Type.CUSTOMER
 });

var salesrep = search.autoJoin({
    fieldId: 'salesrep'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});
search.condition = search.and(
    cond3, search.or(cond1, cond2)
```