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.

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