Version: Paris

API Reference - Server Side Scoped

ServiceNow provides JavaScript APIs for use within scripts running on the ServiceNow platform to deliver common functionality. This reference lists available classes and methods along with parameters, descriptions, and examples to make extending the ServiceNow platform easier.

Please note: The APIs below are intended for scoped applications and may behave differently in the global scope.

GlideQuery

The GlideQuery API is an alternative to GlideRecord to perform CRUD operations on record data from server-side scripts.
The GlideQuery API lets you:

- Use standard JavaScript objects and types for queries and results.
- Quickly diagnose query errors with additional checks and clear error messages.
- Simplify your code by avoiding boiler-plate query patterns.
- Avoid common performance issues without needing deeper knowledge of GlideRecord.

Use the GlideQuery API in scoped or global server-side scripts. This API requires the GlideQuery [com.sn_glidequery] plugin.

Implementation

This API works together with the _____ and _____ APIs in a builder pattern where the method calls chain together, each method building on the returned result of the previous method. Use methods to define the attributes of the query. The methods do not execute until you call a terminal method, a method that returns a query result, allowing you to define the requirements of the query before executing it.

If the query returns a single record, the system wraps the result in an Optional object. If the query returns a stream of records, the system wraps the result in a Stream object. These objects let you manage the result using a set of methods in each API.

For example, here's a script that performs a query on the Task table and groups the records by priority and returns groups with total reassignments greater than four.

```
var query = new global.GlideQuery('task')
    .where('active', true) //Returns new GlideQuery object with a "where" clause.
    .groupBy('priority') //Returns new GlideQuery object with a "group by" clause.
    .aggregate('sum', 'reassignment_count') //Returns new GlideQuery object with a "sum(reassignment_count)" cl
    .having('sum', 'reassignment_count', '>', 4) //Returns new GlideQuery object with a "having reassignment_co
    .select() //Returns a stream of records wrapped in a Stream object.
    .toArray(10); //Terminal method in the Stream class that executes the query and returns the result.
```

Error handling

The GlideQuery API throws an error when your query has a problem, and includes a clear explanation to help guide you. GlideQuery checks for:

- Invalid fields
- Invalid value types for a field
- Invalid values for choice fields
- Invalid query operators

For example, this code sample would throw an error because the queried field does not exist in the table.

```
new global.GlideQuery('task')
    .where('id', '4717dfe5a9fe198100450448b2404c16') // should be 'sys_id'
    .select('description', 'severity')
    .toArray(100);
  // Error: Unable to find field 'id' in table 'task'. Known fields: active, activity_due, ...
```

This code sample would throw an error because the data type of one of the arguments is incorrect.

```
new global.GlideQuery('task')
    .where('priority', 'one') // priority is an integer (should be 1)
    .select('description', 'severity')
    .toArray(100);
  // Error: Unable to match value ['one'] with field 'priority' in table 'task'. Expecting type 'integer'
```

Reuse

Because GlideQuery objects are immutable, you can reuse them later in other parts of your code. For example, this script creates a query and then uses the GlideQuery object later to generate a report.

```
var highPriorityTasks = new global.GlideQuery('task')
    .where('active', true)
    .where('priority', 1);

generateReport(highPriorityTasks);
notifyOwners(highPriorityTasks);
var avgReassignmentCount = highPriorityTasks
    .avg('reassignment_count')
    .orElse(0)
```

Limitations

The GlideQuery API does not support:

- Reading or writing to tables that do not allow access from other scopes.
- Reading encoded queries.
- GlideDate or GlideDateTime objects, which are read as JavaScript strings.
- FX Currency fields.
- Queries with ambiguous conditional logic. For example, the following query is unclear because the system does not know whether to execute `(active = true AND name != null) OR last_name = Luddy` or `active = true AND (name != null OR last_name = Luddy)`.

  ```
  var user = new global.GlideQuery('sys_user')
    .where('active', true)
    .whereNotNull('name')
    .orWhere('last_name', 'Luddy')
    .selectOne()
    .get()
  ```

  See the _____ method to understand how to nest a child query instead.

Note: Because the GlideQuery API converts GlideRecord objects into standard JavaScript objects, it may take longer to execute queries. To reduce performance issues, avoid creating loops that iterate over large numbers of records.

GlideQuery(String table)

Instantiates a GlideQuery object used to build and execute record queries.

Parameters

Name	Type	Description
table	String	Table to query.

Example

This instantiates a query of the User table.

```
var query = new global.GlideQuery('sys_user');
```

aggregate(String aggregateType, String field)

Aggregates a field using a specified aggregation function.

Use this method to build queries that aggregate against multiple fields or use multiple aggregate functions, or if you must use the groupBy() method. If you only want to aggregate against one field with one function, and you don't need to use groupBy(), then use one of these methods instead:

- avg()
- min()
- max()
- count()

Parameters

Name	Type	Description
aggregateType	String	The type of aggregation function to perform. Options include: • `min`: Returns the smallest value of all matching records. • `max`: Returns the largest value of all matching records. • `sum`: Returns the sum of all matching records. • `avg`: Returns the average of all matching records. • `count`: Returns the number of number of matching records.
field	String	Field on which to perform the operation.

Returns

Type	Description
_____	The query object being built.

Example

This performs a query on the Task table that groups the records by priority, adds the numbers in the reassignment count field for each group, and returns groups with total reassignments greater than four.

```
var query = new global.GlideQuery('task')
    .where('active', true) //Returns the GlideQuery object to add more attributes to the query.
    .groupBy('priority') //Returns the GlideQuery object to add more attributes to the query.
    .aggregate('sum', 'reassignment_count') //Returns the GlideQuery object to add more attributes to the query
    .having('sum', 'reassignment_count', '>', 4) //Returns the GlideQuery object to add more attributes to the
    .select() //Returns a stream of records wrapped in a Stream object.
    .toArray(10); //Terminal method in the Stream class that executes the query and returns the result.

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
    "group":{
      "priority":1
    },
    "sum":{
      "reassignment_count":11
    }
  },
  {
    "group":{
      "priority":3
    },
    "sum":{
      "reassignment_count":6
    }
  },
  {
    "group":{
      "priority":5
    },
    "sum":{
      "reassignment_count":5
    }
  }
]
```

avg(String field)

Returns the aggregate average of a given numeric field.

You can only use this method on fields of the following types:

- Integer
- Long
- Floating Point Number
- Double
- Currency

Parameters

Name	Type	Description
field	String	Field on which to perform the operation.

Returns

Type	Description
_____	Object that contains the aggregate average of the given field.

Example

This shows how to return the average number of faults in the cmdb_ci table.

```
var faults = new global.GlideQuery('cmdb_ci')
    .avg('fault_count')
    .orElse(0);

gs.info(JSON.stringify(faults));
```

```
0.0037
```

count()

Returns the number of records that match the query.

Returns

Type	Description
Number	Number of records that match the query.

Example

This returns the number of active records in the User table.

```
var userCount = new global.GlideQuery('sys_user')
    .where('active', true)
    .count();
```

```
612
```

deleteMultiple()

Deletes all records in the table specified by the preceding Where clauses.

Returns

Type	Description
None	

Example

This deletes all active records in the User table where the last name is Jeter.

```
var query = new global.GlideQuery('sys_user')
    .where('active', true)
    .where('last_name', 'Jeter')
    .deleteMultiple();
```

disableAutoSysFields()

Disables updating system fields, or fields with a name that starts with the `sys` prefix, such as sys_created_on, sys_updated_on, and sys_mod_count. Only applies to the specified query.

Returns

Type	Description
	The query object being built.

Example

This adds a record to the task table, but does not set system fields. Without calling this method, the below example would update sys_updated_on, sys_mod_count, and so on.

```
var query = new global.GlideQuery('task')
    .disableAutoSysFields()
    .insert({ description: 'example', priority: 1 });
```

disableWorkflow()

Disables any business rules, flows, workflows, or audit records that would run or be created as the result of the query.

Returns

Type	Description
	The query object being built.

Example

This updates multiple records in the Task table without triggering any automatic business processes.

```
var query = new global.GlideQuery('task')
    .disableWorkflow()
    .where('active', true)
    .updateMultiple({ priority: 1 });
```

forceUpdate()

Forces a database update even when no record changes are made. For example, you can use this method to force a business rule to execute.

Returns

Type	Description
	The query object being built.

Example

This force updates Task records with a certain sys_id.

```
var forceUpdate = new global.GlideQuery('task')
    .forceUpdate()
    .where('sys_id', taskId)
    .update()
```

get(String key, Array selectedFields)

Returns a single record from the query.

Parameters

Name	Type	Description
key	String	Sys_id of the record to return.
selectedFields	Array	Optional. Additional fields to return in the result. Default: The system always returns the sys_id.

Returns

Type	Description
	Object used to interact with a single record.

Example

that returns a record based on sys_id.

```
var user = new global.GlideQuery('sys_user')
    .get('5137153cc611227c000bbd1bd8cd2005', ['first_name', 'last_name']) //Returns an Optional object.
    .orElse({ first_name: 'Default', last_name: 'User' }); //Method in the Optional class to return a default v

gs.info(JSON.stringify(user, null, 2));
```

```
{
   "sys_id":"5137153cc611227c000bbd1bd8cd2005",
   "first_name":"Fred",
   "last_name":"Luddy"
}
```

getBy(Object keyValues, Array selectedFields)

Returns an Optional object containing a single record based on a set of name-value pairs to query by. Assumes the '=' operator for each name-value pair.

Parameters

Name	Type	Description
keyValues	Object	Object where the keys are the name of the fields, and the values are the values to query for.
selectedFields	Array	Optional. Additional fields to return in the result. Default: The system always returns the sys_id.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

that returns a record by querying for a user's name.

```
var user = new global.GlideQuery('sys_user')
    .getBy({
        first_name: 'Fred',
        last_name: 'Luddy'
    }, ['first_name', 'last_name', 'city', 'active']) // select first_name, last_name, city, active
    .orElse({
        first_name: 'Nobody',
        last_name: 'Found',
        city: 'Nowhere',
        active: false
    });

gs.info(JSON.stringify(user, null, 2));
```

```
{
   "first_name":"Fred",
   "last_name":"Luddy",
   "city":null,
   "active":true,
   "sys_id":"5137153cc611227c000bbd1bd8cd2005"
}
```

groupBy(String fields)

Groups the query results by a designated field or fields.

You must use this method with the aggregate() method.

Parameters

Name	Type	Description
fields	String or Array of Strings	Field or fields to group the results by.

Returns

Type	Description
	The query object being built.

Example

This performs a query on the Task table that groups the records by priority, adds the numbers in the reassignment count field for each group, and returns groups with total reassignments greater than four.

```
var query = new global.GlideQuery('task')
    .where('active', true) //Returns the GlideQuery object to add more attributes to the query.
    .groupBy('priority') //Returns the GlideQuery object to add more attributes to the query.
    .aggregate('sum', 'reassignment_count') //Returns the GlideQuery object to add more attributes to the query
    .having('sum', 'reassignment_count', '>', 4) //Returns the GlideQuery object to add more attributes to the
    .select() //Returns a stream of records wrapped in a Stream object.
    .toArray(10); //Terminal method in the Stream class that executes the query and returns the result.

gs.info(JSON.stringify(query, null, 2));
```

```
[
   {
      "group":{
         "priority":1
      },
      "sum":{
         "reassignment_count":11
      }
   },
   {
      "group":{
         "priority":3
      },
      "sum":{
         "reassignment_count":6
      }
   },
   {
      "group":{
         "priority":5
      },
      "sum":{
         "reassignment_count":5
      }
   }
]
```

having(String aggregateType, String field, String operator, Number value)

Filters aggregate groups so that you can display only groups of results that match a specified condition.

Must use this method with the aggregate() or groupBy() methods.

Parameters

Name	Type	Description
aggregateType	String	The type of aggregation function to perform. Options include: • `min`: Returns the smallest value of all matching records. • `max`: Returns the largest value of all matching records. • `sum`: Returns the sum of all matching records. • `avg`: Returns the average of all matching records. • `count`: Returns the number of number of matching records.
field	String	Field on which to perform the operation.
operator	String	Numeric operator to use in the operation. Options include: • `>`: Greater than. • `<`: Less than. • `>=`: Greater than or equal to. • `<=`: Less than or equal to. • `=`: Equal to. • `!=`: Not equal to.
value	Number	Number value to use in the operation.

Returns

Type	Description
	The query object being built.

Example

This performs a query on the Task table that groups the records by priority, adds the numbers in the reassignment count field for each group, and returns groups with total reassignments greater than four.

```
var query = new global.GlideQuery('task')
    .where('active', true) //Returns the GlideQuery object to add more attributes to the query.
    .groupBy('priority') //Returns the GlideQuery object to add more attributes to the query.
    .aggregate('sum', 'reassignment_count') //Returns the GlideQuery object to add more attributes to the query
    .having('sum', 'reassignment_count', '>', 4) //Returns the GlideQuery object to add more attributes to the
    .select() //Returns a stream of records wrapped in a Stream object.
    .toArray(10); //Terminal method in the Stream class that executes the query and returns the result.

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
     "group":{
        "priority":1
     },
     "sum":{
        "reassignment_count":11
     }
  },
  {
     "group":{
        "priority":3
     },
     "sum":{
        "reassignment_count":6
     }
  },
  {
     "group":{
        "priority":5
     },
     "sum":{
        "reassignment_count":5
     }
  }
]
```

insert(Object keyValues, Object selectedFields)

Inserts a record and returns an Optional object containing the record.

Parameters

Name	Type	Description
keyValues	Object	Object containing name-value pairs to insert into the record. Unspecified fields will be null.
selectedFields	Array	Optional. Additional fields to return in the result. Default: The system always returns the sys_id.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

This shows how to insert a record based on a user's first and last name.

```
var fred = new global.GlideQuery('sys_user')
    .insert({ first_name: 'Fred', last_name: 'Luddy' })
    .get();

gs.info(JSON.stringify(fred, null, 2));
```

```
{
   "sys_id":"cf16eed0e82a9010f8778bda83d255d2",
   "first_name":"Fred",
   "last_name":"Luddy"
}
```

insertOrUpdate(Object changes, Object selectedFields)

Updates an existing record, or inserts a new record if one does not already exist.

Parameters

Name	Type	Description
changes	Object	Object containing name-value pairs to update or insert into the record.
selectedFields	Array	Optional. Additional fields to return in the result. Default: The system always returns the sys_id.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

This shows how to insert a new record that does not already exist in the system.

```
// insert a new record
var user = new GlideQuery('sys_user')
    .insertOrUpdate({
        first_name: 'George',
        last_name: 'Griffey'
    })
    .orElse(null);
```

Example

This shows how to update an existing record.

```
// update existing record
var user = new global.GlideQuery('sys_user')
    .insertOrUpdate({
        sys_id: '2d0efd6c73662300bb513198caf6a72e',
        first_name: 'George',
        last_name: 'Griffey' })
    .orElse(null);
```

limit(Number limit)

Limits the number of records returned in a query.

Parameters

Name	Type	Description
limit	Number	Number of records to return.

Returns

Type	Description
	The query object being built.

Example

This shows how to limit the results returned to five records.

```
var incidents = new global.GlideQuery('incident')
    .limit(5)
    .select('priority', 'description')
    .toArray(100);

gs.info(JSON.stringify(incidents, null, 2));
```

```
[
   {
      "priority":3,
      "description":"I am unable to connect to the email server. It appears to be down.",
      "sys_id":"1c741bd70b2322007518478d83673af3"
   },
   {
      "priority":3,
      "description":"My computer is not detecting the headphone device. It could be an issue with the USB port.",
      "sys_id":"1c832706732023002728660c4cf6a7b9"
   },
   {
      "priority":1,
      "description":"I can't remember my password and need to log in. Can someone reset my password asap? I am blocked on several urgent it
      "sys_id":"46b66a40a9fe198101f243dfbc79033d"
   },
   {
      "priority":4,
      "description":"Currently running 10GR1 and need to upgrade to 10GR2.",
      "sys_id":"46b9490da9fe1981003c938dab89bda3"
   },
   {
      "priority":3,
      "description":"I'm replacing my old phone with a Blackberry and require assistance to get it set up. I'd like to get the files and co
      "sys_id":"46c03489a9fe19810148cd5b8cbf501e"
   }
]
```

max(String field)

Returns the aggregate maximum of a given field.

Parameters

Name	Type	Description
field	String	Field on which to perform the operation.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

This shows how to return the maximum value, or highest alphanumeric value, of a given field.

```
var name = new global.GlideQuery('sys_user')
    .max('last_name')
    .orElse('');

gs.info(JSON.stringify(name));
```

```
"Zortman"
```

min(String field)

Returns the aggregate minimum of a given field.

Parameters

Name	Type	Description
field	String	Field on which to perform the operation.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

This shows how to return the minimum value, or lowest alphanumeric value, of a given field.

```
var name = new global.GlideQuery('sys_user')
    .min('last_name')
    .orElse('');

gs.info(JSON.stringify(name));
```

```
"Abel"
```

orderBy(String fields)

Orders the returned result in ascending order by a given field.

Parameters

Name	Type	Description
fields	String	Comma-delimited fields to order the result by in ascending order.

Returns

Type	Description
_____	The query object being built.

Example

This shows how to order results in ascending order by record number.

```
var query = new global.GlideQuery('incident')
    .orderBy('number')
    .limit(5)
    .select('priority', 'description') //Returns a stream of records wrapped in a Stream object.
    .toArray(100); //Terminal method in the Stream class that executes the query and returns the result.

gs.info(JSON.stringify(query, null, 2));
```


```
[
   {
      "priority":1,
      "description":"User can't access email on mail.company.com.\n\t\t",
      "sys_id":"9c573169c611228700193229fff72400"
   },
   {
      "priority":1,
      "description":"User can't get to any of his files on the file server.",
      "sys_id":"9d385017c611228701d22104cc95c371"
   },
   {
      "priority":1,
      "description":"I just moved from floor 2 to floor 3 and my laptop cannot connect to any wireless network.",
      "sys_id":"e8caedcbc0a80164017df472f39eaed1"
   },
   {
      "priority":1,
      "description":"User forgot their email password.",
      "sys_id":"9d3c1197c611228701cd1d94bc32d76d"
   },
   {
      "priority":1,
      "description":"Watcher daemon detected that the CPU was 100% busy for more than 10 minutes",
      "sys_id":"e8e875b0c0a80164009dc852b4d677d5"
   }
```


orderByDesc(String fieldOrAggregate, String field)

Orders the returned result in descending order by a given field.

Parameters

Name	Type	Description
fieldOrAggregate	String	If the query does not use the aggregate() method, pass the field to order the results by. If the query uses the aggregate() method, pass the type of aggregation function to perform. Options include: • `min`: Returns the smallest value of all matching records. • `max`: Returns the largest value of all matching records. • `sum`: Returns the sum of all matching records. • `avg`: Returns the average of all matching records. • `count`: Returns the number of number of matching records.
field	String	Optional. Field to order the result by in descending order. Required for queries using the aggregate() method.

Returns

Type	Description
	The query object being built.

Example

This shows how to order the result in descending order by number.

```
var query = new global.GlideQuery('incident')
    .orderByDesc('number')
    .select('number', 'description')
    .limit(5)
    .toArray(100);

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
     "number":"INC0010112",
     "description":null,
     "sys_id":"552c48888c033300964f4932b03eb092"
  },
  {
     "number":"INC0010111",
     "description":null,
     "sys_id":"a83820b58f723300e7e16c7827bdeed2"
  },
  {
     "number":"INC0009009",
     "description":"Unable to access the shared folder. Please provide access.",
     "sys_id":"57af7aec73d423002728660c4cf6a71c"
  },
  {
     "number":"INC0009005",
     "description":"Unable to send or receive emails.",
     "sys_id":"ed92e8d173d023002728660c4cf6a7bc"
  },
  {
     "number":"INC0009004",
     "description":"While launching the defect tracking base URL, it is redirecting to an error page.",
     "sys_id":"e329de99731423002728660c4cf6a73c"
  }
]
```

Example

This shows how to order an aggregate result by the sum of child incidents.

```
var aggQuery = new GlideQuery('incident')
    .aggregate('sum', 'child_incidents')
    .groupBy('category')
    .orderByDesc('sum', 'child_incidents')
    .select()
    .toArray(100);

gs.info(JSON.stringify(aggQuery, null, 2));
```

```
[
   {
      "group":{
         "category":"hardware"
      },
      "sum":{
         "child_incidents":2
      }
   },
   {
      "group":{
         "category":"inquiry"
      },
      "sum":{
         "child_incidents":1
      }
   },
   {
      "group":{
         "category":"software"
      },
      "sum":{
         "child_incidents":0
      }
   },
   {
      "group":{
         "category":""
      },
      "sum":{
         "child_incidents":null
      }
   },
   {
      "group":{
         "category":"database"
      },
      "sum":{
         "child_incidents":null
      }
   },
   {
      "group":{
         "category":"network"
      },
      "sum":{
         "child_incidents":null
      }
   }
]
```


orWhere(String fieldOrQuery, String operator, Any value)

Adds an OR clause to a query that returns values based on a given condition.

Note: Precede this method with the where(), whereNull(), or whereNotNull() methods.

Parameters

Name	Type	Description
fieldOrQuery	String or _____ _____ _____	Field or another GlideQuery object used in the where clause. If passing a field, you can dot-walk to a desired value. For example, `'company.name'`.
operator	String	Optional. Operator used in the OR clause. If you do not pass an argument, the system uses the = operator. You do not need to include a placeholder value.
value	Any	Value used in the OR clause.

Returns

Type	Description
_____ _____	The query object being built.

Example

This shows how to add a simple OR clause to a query.

```
var query = new global.GlideQuery('sys_user')
   .where('failed_attempts', '>', 0)
   .orWhere('last_login', '<', '2019-04-15')
   .select()
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
    "sys_id":"005d500b536073005e0addeeff7b12f4"
  },
  {
    "sys_id":"30ad318577ab2300454792718a10619e"
  },
  {
    "sys_id":"3883f4c0730123002728660c4cf6a754"
  },
  {
    "sys_id":"3988a3ca732023002728660c4cf6a757"
  },
  {
    "sys_id":"4ac73ecd738123002728660c4cf6a72c"
  },
  {
    "sys_id":"8ff5b254b33213005e3de13516a8dcf7"
  },
  {
    "sys_id":"d999e5fc77e72300454792718a10611d"
  }
]
```

Example

This shows how to add a `orWhere` clause that contains a separate query.

```
// active = true OR (title = 'Vice President' AND state = 'CA')
var query = new GlideQuery('sys_user')
   .where('active', true)
   .orWhere(new GlideQuery()
       .where('title', 'Vice President')
       .where('state', 'CA'))
   .select('name')
   .limit(5)
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
     "name":"survey user",
     "sys_id":"005d500b536073005e0addeeff7b12f4"
  },
  {
     "name":"Lucius Bagnoli",
     "sys_id":"02826bf03710200044e0bfc8bcbe5d3f"
  },
  {
     "name":"Jimmie Barninger",
     "sys_id":"02826bf03710200044e0bfc8bcbe5d55"
  },
  {
     "name":"Melinda Carleton",
     "sys_id":"02826bf03710200044e0bfc8bcbe5d5e"
  },
  {
     "name":"Jewel Agresta",
     "sys_id":"02826bf03710200044e0bfc8bcbe5d64"
  }
]
```

orWhereNotNull(String field)

Adds an OR clause that returns records that do not contain a null value in a given field.

Note: Precede this method with the where(), whereNull(), or whereNotNull() methods.

Parameters

Name	Type	Description
field	String	Field used in the query.

Returns

Type	Description
	The query object being built.

Example

This shows how to query the User table and return results where the first and last names are not null.

```
var query = new global.GlideQuery('sys_user')
   .whereNotNull('first_name')
   .orWhereNotNull('last_name')
   .select()
   .limit(5)
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
    "sys_id":"005d500b536073005e0addeeff7b12f4"
  },
  {
    "sys_id":"02826bf03710200044e0bfc8bcbe5d3f"
  },
  {
    "sys_id":"02826bf03710200044e0bfc8bcbe5d55"
  },
  {
    "sys_id":"02826bf03710200044e0bfc8bcbe5d5e"
  },
  {
    "sys_id":"02826bf03710200044e0bfc8bcbe5d64"
  }
]
```

orWhereNull(String field)

Adds an OR clause to a query that returns records that contain a null value in a given field.

Note: Precede this method with the where(), whereNull(), or whereNotNull() methods.

Parameters

Name	Type	Description
field	String	Field used in the query.

Returns

Type	Description
_____	The query object being built.

Example

This shows how to query the User table and return records where the first or last names are null.

```
var query = new global.GlideQuery('sys_user')
   .whereNull('last_name')
   .orWhereNull('first_name')
   .select()
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
   {
      "sys_id":"1195569be81a1010f8778bda83d25585"
   },
   {
      "sys_id":"5136503cc611227c0183e96598c4f706"
   }
]
```

select(String fields)

Returns the results of the query as a Stream object containing specified fields.

Note: Use a terminal method in the Stream class to get the result of the query. For more information, see _____.

You can append a flag to a field name to return the field's metadata instead of the field's value. For example, using the field name `company$DISPLAY` returns the display value of a company field. Possible flags include:

- `DISPLAY`: Returns the display value of a field.
- `CURRENCY_CODE`: Returns the currency code of a currency field. For example, `USD`.
- `CURRENCY_DISPLAY`: Returns the currency display value of a currency field. For example, `¥123.45`.
- `CURRENCY_STRING`: Returns the currency string of a currency field. For example, `JPY;123.45`.

Parameters

Name	Type	Description

fields	String or Array of Strings	Optional. Fields to display in the result. You can provide any number of fields as arguments, dot-walk to a desired value, or use a flag. For example:
		`select('first_name', 'location.city', 'company$DISPLAY');`
		or
		`select(['first_name', 'location.city', 'company$DISPLAY']);`
		Default: The system always returns the sys_id.

Returns

Type	Description
_____	Object used to interact with a stream of items such as records.

Example

This shows how to select fields to display from the query and use `$DISPLAY` to return the display value of a field.

```
var stream = new global.GlideQuery('sys_user')
   .select('first_name', 'last_name', 'company$DISPLAY')
   .toArray(5);

gs.info(JSON.stringify(stream, null, 2));
```

```
[
   {
      "first_name":"survey",
      "last_name":"user",
      "company$DISPLAY":"",
      "sys_id":"005d500b536073005e0addeeff7b12f4"
   },
   {
      "first_name":"Lucius",
      "last_name":"Bagnoli",
      "company$DISPLAY":"ACME Japan",
      "sys_id":"02826bf03710200044e0bfc8bcbe5d3f"
   },
   {
      "first_name":"Jimmie",
      "last_name":"Barninger",
      "company$DISPLAY":"ACME South America",
      "sys_id":"02826bf03710200044e0bfc8bcbe5d55"
   },
   {
      "first_name":"Melinda",
      "last_name":"Carleton",
      "company$DISPLAY":"ACME UK",
      "sys_id":"02826bf03710200044e0bfc8bcbe5d5e"
   },
   {
      "first_name":"Jewel",
      "last_name":"Agresta",
      "company$DISPLAY":"ACME UK",
      "sys_id":"02826bf03710200044e0bfc8bcbe5d64"
   }
]
```

selectOne(String fields)

Returns the result of the query as an Optional object containing specified fields.

Use this method when returning a single record, or to test if a record exists. If returning multiple records, use the select() method to return a Stream object.

You can append a flag to a field name to return the field's metadata instead of the field's value. For example, using the field name `company$DISPLAY` returns the display value of a company field. Possible flags include:

- `DISPLAY`: Returns the display value of a field.
- `CURRENCY_CODE`: Returns the currency code of a currency field. For example, `USD`.
- `CURRENCY_DISPLAY`: Returns the currency display value of a currency field. For example, `¥123.45`.
- `CURRENCY_STRING`: Returns the currency string of a currency field. For example, `JPY;123.45`.

Parameters

Name	Type	Description

fields	String or Array of Strings	Optional. Fields to display in the result. You can provide any number of fields as arguments, dot-walk to a desired value, or use a flag. For example:
		`selectOne('first_name', 'location.city', 'company$DISPLAY');`
		or
		`selectOne(['first_name', 'location.city', 'company$DISPLAY']);`
		Default: The system always returns the sys_id.

Returns

Type	Description
	Object used to interact with a single record.

Example

This shows how to return a single record as an Optional object and display specified fields.

```
var user = new global.GlideQuery('sys_user')
    .where('zip', '12345')
    .whereNotNull('last_name')
    .selectOne('first_name', 'last_name', 'company$DISPLAY')
    .get();

gs.info(JSON.stringify(user, null, 2));
```

```
{
   "first_name":"Abel",
   "last_name":"Tuter",
   "company$DISPLAY":"ACME South America",
   "sys_id":"62826bf03710200044e0bfc8bcbe5df1"
}
```

sum(String field)

Returns the aggregate sum of a given numeric field.

You can only use this method on fields of the following types:

- Integer
- Long
- Floating Point Number
- Double
- Currency

Parameters

Name	Type	Description
field	String	Field on which to perform the operation.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

This shows how to return the sum of all faults in the cmdb_ci table.

```
var totalFaults = new global.GlideQuery('cmdb_ci')
    .sum('fault_count')
    .orElse(0);

gs.info(JSON.stringify(totalFaults));
```

```
10
```

toGlideRecord()

Returns a GlideRecord object that represents the current query. Returns a GlideAggregate object if the query uses the GlideQuery.aggregate() method.

After transforming the query, use the query() method in the _____ _____ or _____ _____ classes to query the database.

Returns

Type	Description
or	GlideRecord object that contains the query. If you used the GlideQuery.aggregate() method, then the method returns a GlideAggregate object instead.

Example

This shows how to transform a GlideQuery object into a GlideRecord.

```
var userGr = new global.GlideQuery('sys_user')
    .where('active', true)
    .whereNotNull('first_name')
    .limit(10)
    .toGlideRecord();
userGr.query();
```

update(Object changes, Object selectedFields)

Updates an existing record that matches the defined conditions.

Before using this method, call the where() method to specify the conditions that a record must meet to be updated.

Parameters

Name	Type	Description
changes	Object	Object containing name-value pairs to update in the record. Names must match fields in the table.
selectedFields	Array	Optional. Additional fields to return in the result. Default: The system always returns the sys_id.

Returns

Type	Description
_____	Object used to interact with a single record.

Example

This shows how to update a record with new values.

```
var updateRecord = new global.GlideQuery('sys_user')
    .where('sys_id', '0a826bf03710200044e0bfc8bcbe5d7a')
    .update({ city: 'Los Angeles' });
```

updateMultiple(Object changes)

Updates all existing records that match the defined conditions. Returns the number of records updated.

Before using this method, call the where() method to specify the conditions that the records must meet to be updated.

Parameters

Name	Type	Description
changes	Object	Object containing name-value pairs to update in the record. Names must match fields in the table.

Returns

Type	Description
Object	Object containing the number of records that were updated. Keys include: • `rowCount`: Number of rows updated in the table.

Example

This shows how to update any records that fit a defined criteria.

```
var update = new global.GlideQuery('sys_user')
    .where('active', false)
    .where('last_name', 'Griffey')
    .updateMultiple({ active: true });

gs.info(JSON.stringify(update));
```

```
{"rowCount":1}
```

where(String fieldOrQuery, String operator, Any value)

Adds a Where clause to the query that returns values based on a given condition.

Note: Do not precede this method with the orWhere(), orWhereNull(), or orWhereNotNull() methods.

Parameters

Name	Type	Description
fieldOrQuery	String or _____	Field or another GlideQuery object used in the where clause. If passing a field, you can dot-walk to a desired value. For example, `'company.name'`.
operator	String	Optional. Operator used in the where clause. If you do not pass an argument, the system uses the = operator.
value	Any	Value used in the where clause.

Returns

Type	Description
	The query object being built.

Example

This shows how to return records from the User table where active is true and the user last logged on after a specified date.

```
var query = new global.GlideQuery('sys_user')
   .where('active', true)
   .where('last_login', '>', '2016-04-15')
   .limit(5)
   .select()
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
  {
    "sys_id":"b0f31e5673500010c2e7660c4cf6a711"
  },
  {
    "sys_id":"8ff5b254b33213005e3de13516a8dcf7"
  },
  {
    "sys_id":"d999e5fc77e72300454792718a10611d"
  },
  {
    "sys_id":"30ad318577ab2300454792718a10619e"
  },
  {
    "sys_id":"3883f4c0730123002728660c4cf6a754"
  }
]
```

Example

This shows how to return records from the Incident table where active is true and where either the priority or the severity is 1.

```
// active = true AND (priority = 1 OR severity = 1)
var query = new GlideQuery('incident')
    .where('active', true)
    .where(new GlideQuery()
        .where('priority', 1)
        .orWhere('severity', 1))
   .limit(5)
   .select()
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
   {
      "sys_id":"b0f31e5673500010c2e7660c4cf6a711"
   },
   {
      "sys_id":"8ff5b254b33213005e3de13516a8dcf7"
   },
   {
      "sys_id":"d999e5fc77e72300454792718a10611d"
   },
   {
      "sys_id":"30ad318577ab2300454792718a10619e"
   },
   {
      "sys_id":"3883f4c0730123002728660c4cf6a754"
   }
]
```

whereNotNull(String field)

Returns records that do not contain a null value in a given field.

Note: Do not precede this method with the orWhere(), orWhereNull(), or orWhereNotNull() methods.

Parameters

Name	Type	Description
field	String	Field used in the query.

Returns

Type	Description
_____	The query object being built.

Example

This shows how to query the User table and return results where the first_name field is not null.

```
var query = new global.GlideQuery('sys_user')
   .whereNotNull('first_name')
   .select()
   .limit(5)
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
   {
      "sys_id":"005d500b536073005e0addeeff7b12f4"
   },
   {
      "sys_id":"02826bf03710200044e0bfc8bcbe5d3f"
   },
   {
      "sys_id":"02826bf03710200044e0bfc8bcbe5d55"
   },
   {
      "sys_id":"02826bf03710200044e0bfc8bcbe5d5e"
   },
   {
      "sys_id":"02826bf03710200044e0bfc8bcbe5d64"
   }
]
```

whereNull(String field)

Returns records that contain a null value in a given field.

Note: Do not precede this method with the orWhere(), orWhereNull(), or orWhereNotNull() methods.

Parameters

Name	Type	Description
field	String	Field used in the query.

Returns

Type	Description
_____	The query object being built.

Example

This shows how to query the User table and return records where the first or last names are null.

```
var query = new global.GlideQuery('sys_user')
   .whereNull('last_name')
   .orWhereNull('first_name')
   .select()
   .toArray(100)

gs.info(JSON.stringify(query, null, 2));
```

```
[
   {
      "sys_id":"1195569be81a1010f8778bda83d25585"
   },
   {
      "sys_id":"5136503cc611227c0183e96598c4f706"
   }
]
```

withAcls()

Executes the query using the GlideRecordSecure API to securely query the database while honoring ACLs.

Returns

Type	Description
	The query object being built.

Example

This shows how to execute a secure query using ACLs.

```
var users = new global.GlideQuery('sys_user')
    .withAcls()
    .limit(5)
    .orderByDesc('first_name')
    .select('first_name')
    .toArray(100);

gs.info(JSON.stringify(users, null, 2));
```

```
[
   {
      "first_name":"Zane",
      "sys_id":"16826bf03710200044e0bfc8bcbe5dbc"
   },
   {
      "first_name":"Zackary",
      "sys_id":"8a826bf03710200044e0bfc8bcbe5d69"
   },
   {
      "first_name":"Yvette",
      "sys_id":"4e826bf03710200044e0bfc8bcbe5d57"
   },
   {
      "first_name":"Winnie",
      "sys_id":"f682abf03710200044e0bfc8bcbe5d1d"
   },
   {
      "first_name":"Wilmer",
      "sys_id":"42826bf03710200044e0bfc8bcbe5d7b"
   }
]
```