Glide Reference

Glide Methods and Usage

PDF generated using the open source mwlib toolkit. See http://code.pediapress.com/ for more information.
PDF generated at: Wed, 29 Nov 2017 20:11:26 PST

Server Side

GlideRecord

Note: *This article applies to Fuji. For more current information, see GlideRecord* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

GlideRecord is a special Java class (GlideRecord.java) that can be used in JavaScript exactly as if it was a native JavaScript class.

GlideRecord

- is used for database operations instead of writing SQL queries.
- is an object that contains zero or more records from one table. Another way to say this is that a GlideRecord is an ordered list.

Note: *The global Glide APIs are not available in scoped scripts. There are scoped Glide APIs for use in scoped scripts. The scoped Glide APIs provide a subset of the global Glide API methods. For information on scoped applications see Application Scope, scripting in scoped applications, and the list of scoped APIs.*

Use this API to:

- Query
- Get
- Set
- Update
- Insert
- Delete

A GlideRecord contains both records (rows) and fields (columns). The field names are the same as the underlying database column names.

For more information on available JavaScript operators and additional GlideRecord examples, see Using GlideRecord to Query Tables.

Query

Query Method Summary

Return Value	Details
QueryCondition	**addActiveQuery**() Adds a filter to return active records.
void	**addDomainQuery**(`Object o`) Changes the domain used for the query from the user's domain to the domain of the provided GlideRecord.
void	**addEncodedQuery**(`String query`) Adds an encoded query. Adds an encoded query to the other queries that may have been set.
QueryCondition	**addInactiveQuery**() Adds a filter to return inactive records.
QueryCondition	**addJoinQuery**(`joinTable, [primaryField], [joinTableField]`) Adds a filter to return records based on a relationship in a related table.
QueryCondition	**addNotNullQuery**(`String fieldName`) Adds a filter to return records where the specified field is not null.
Query Condition	**addNullQuery**(`String fieldName`) Adds a filter to return records where the specified field is null.
QueryCondition	**addOrCondition**(`String fieldName, Object operator, Object value`) Adds an alternative filter to an existing query to return records based on 1, 2 or 3 arguments. • 1 argument adds an encoded query string. • 2 arguments return records where the field is equal to the value (or is in a list of values). • 3 arguments return records where the field meets the specified condition (field, operator and value).
QueryCondition	**addQuery**(`String fieldName, Object operator, Object value`) Adds a filter to return records based on 1, 2 or 3 arguments. • 1 argument adds an encoded query string. • 2 arguments return records where the field is equal to the value (or is in a list of values). • 3 arguments return records where the field meets the specified condition (field, operator and value).
boolean	**canCreate**() Determines if the Access Control Rules (which includes the user's role) permit inserting new records in this table.
boolean	**canDelete**() Determines if the Access Control Rules (which includes the user's role) permit deletion of records in this table.
boolean	**canRead**() Determines if the Access Control Rules (which includes the user's role) permit reading this table.
boolean	**canWrite**() Determines if the Access Control Rules (which includes the user's role) permit updates to records in this table.
boolean	**changes**() Determines whether any of the fields in the record have changed.
boolean	**find**(`columnName, value`) Returns true if any record has a matching value in the specified column. If found, it also moves to the first record that matches, essentially executing next() until the record is returned.
boolean	**hasAttachments**() Determines if the current record has any attachments.
boolean	**hasNext**() Determines if there are any more records in the GlideRecord.

boolean	**instanceOf**(`String className`) Checks a table for a type/class of record.
boolean	**isNewRecord**() Determines whether the current record has been inserted into the database.
boolean	**isValid**() Determines whether the table exists or not.
boolean	**isValidField**(`String columnName`) Determines if the given field is defined in the current table.
boolean	**isValidRecord**() Determine if there is another record in the GlideRecord.
boolean	**next**() Moves to the next record in the GlideRecord.
boolean	**_next**() Provides the same functionality as `next`, intended to be used in cases where the GlideRecord has a column named **next.**
String	**operation**() Retrieves the current operation being performed, such as insert, update, delete, etc.
void	**orderBy**(`String name`) Specifies an orderBy column (this may be called more than once to order by multiple columns).
void	**orderByDesc**(`String name`) Specifies a decending orderBy column.
void	**query**(`Object field, Object value`) Runs the query against the table based on the specified filters by addQuery and addEncodedQuery. This will query the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but a name/value pair can be specified.
void	**queryNoDomain**(`Object field, Object value`) Used only in domain separated instances. Performs the same function as `query();`, but turns off domain processing.
void	**_query**(`Object field, Object value`) Identical to `query();`, intended to be used on tables where there is a column named **'query**, which would interfere with using `query();`.
void	**restoreLocation**() Sets the current record to be the record that was saved with `saveLocation()`. If saveLocation has not been called yet, the current record is set to be the first record of the GlideRecord.
void	**saveLocation**() Save the current row number so that we can get back to this location using `restoreLocation()`.

Get

Get Method Summary

Return Value	Details
boolean	**get**(`Object name, Object value`) Defines a GlideRecord based on the specified expression of 'name = value'. If value is not specified, then the expression used is 'sys_id = name'. This method is expected to be used to query for single records, so a 'next' operation on the GlideRecord is performed by this method before returning.
String	**getAttribute**(`String attribute`) Retrieves the attributes on the field in question from the dictionary.
String	**getClassDisplayValue**() Retrieves the label for the table.
String	**getDisplayValue**() Retrieves the display value for the current record.
ElementDescriptor	**getED**() Retrieves the element's descriptor.
GlideElement	**getElement**(`String columnName`) Retrieves the GlideElement for a specified field.
String	**getEncodedQuery**() Retrieves the encoded query as a string.
String	**getEscapedDisplayValue**() Retrieves the field value of the current record and escapes it for use in Jelly scripts.
ArrayList	**getFields**() Retrieves an array of fields in the current record.
String	**getLabel**() Retrieves the appropriate label for a field.
String	**getLink**(`boolean noStack`) Retrieves the link for the current record.
int	**getLocation**() Retrieves the current row number.
string	**getPlural**() Retrieves the plural label of the GlideRecord table.
String	**getRecordClassName**() Retrieves the class name for the current record.
HashMap	**getRelatedLists**() Retrieves a list of names and display values of tables that **refer** to the current record.
HashMap	**getRelatedTables**() Retrieves a list of names and display values of tables that are **referred to** by the current record.
int	**getRowCount**() Retrieves the number of rows in the GlideRecord.
int	**getRowNumber**() Retrieves the row number set by saveLocation() or setLocation().
String	**getTableName**() Retrieves the table name associated with this GlideRecord.

String	**getValue**(`String fieldName`) Retrieves the string value of an underlying element in a field.

Set

Set Method Summary	
Return Value	**Details**
void	**autoSysFields**(`Boolean e`) Enables or disables the update to the fields sys_updated_by, sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on. This is often used for manually updating field values on a record while leaving historical information unchanged. **Note:** This is not available for scoped apps, starting with the Fuji release. See the Scoped GlideRecord API Reference for a list of what APIs are available for scoped apps.
void	**setAbortAction**(`boolean b`) Sets a flag to indicate if the next database action (insert, update, delete) is to be aborted.
void	**setDisplayValue**(`String name, Object value`) Sets the field name to the specified display value. • For a reference field this is the display value for the table. • For a date/time field this is the time in the caller's current timezone.
void	**setForceUpdate**(`boolean e`) Updates the record even if fields have not been changed.
void	**setLimit**(`int limit`) Sets the limit for how many records in the GlideRecord.
void	**setLocation**(`int Number`) Sets the current row location.
void	**setNewGuid**() Generates a new GUID and sets it as the unique id for the current record.
void	**setNewGuidValue**(`String guid`) Generates a new GUID and sets it as the unique id for the current record, when inserting a new record.
void	**setQueryReferences**(`boolean queryReferences`) Enables or disables using the reference field's display name when querying a reference field.
void	**setUseEngines**(`boolean e`) Disable or enable the running of any engines (approval rules / assignment rules).
void	**setValue**(`String fieldName, Object value`) Sets the specified field to the specified value. Normally a script would do a gr.category = value. However, if the element name is a variable then gr.setValue(elementName, value) can be used.
return	**setWorkflow**(`boolean e`) Enables or disables the running of business rules that might normally be triggered by subsequent actions.

Update

Update Method Summary	
Return Value	**Details**
void	**applyTemplate**(`String template`) Apply a template record (from sys_templates) to the current record. If the specified template is not found, no action is taken.
string	**update**(`Object reason`) Updates the GlideRecord with any changes that have been made. If the record does not already exist, it is inserted.
string	**updateWithReferences**(`Object reason`) Updates a record and also inserts or updates any related records with the information provided.

Insert

Insert Method Summary	
Return Value	**Details**
void	**initialize**() Creates an empty record suitable for population before an insert.
string	**insert**() Insert a new record using the field values that have been set for the current record.
string	**insertWithReferences**() Inserts a new record and also inserts or updates any related records with the information provided.
void	**newRecord**() Creates a GlideRecord, set the default values for the fields and assign a unique id to the record. See Also: initialize().

Delete

Delete Method Summary	
Return Value	**Details**
void	**deleteMultiple**() Delete multiple records according to the current "where" clause. Does not delete attachments.
boolean	**deleteRecord**() Delete a single record.

Query Methods

addActiveQuery

public QueryCondition addActiveQuery()

Adds a filter to return active records.

Returns:

`QueryCondition` - filter to return active records.

Example:

```
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.query();
```

addEncodedQuery

public void addEncodedQuery(`String query`)

Adds an encoded query. Adds an encoded query to the other queries that may have been set.

Parameters:

`query` - An encoded query string to add to the record.

Example:

```
var queryString = "priority=1^ORpriority=2";
 var gr = new GlideRecord('incident');

 gr.addEncodedQuery(queryString);
 gr.query();
 while (gr.next()) {
   gs.addInfoMessage(gr.number);
 }
```

Use the breadcrumbs and filters to generate encoded query strings.

addDomainQuery

public void addDomainQuery(`Object o`)

Changes the domain used for the query from the user's domain to the domain of the provided GlideRecord.

Parameters:

`o` - A GlideRecord from which to draw the domain.

Example:

```
//This example requires the Domain plugin be active, the Group table is
 the specified Domain table,
//and the ITIL user is in the Database Atlanta domain
//From any domain (using queryNoDomain()) look up the incidents that an
 ITIL user can only see
//who is in the Database Atlanta domain, should expect all incidents
with the global or the
//Database Atlanta domain specified.
var domain = new GlideRecord('sys_user');
domain.addQuery('user_name', 'itil');
domain.queryNoDomain();
if (domain.next()) {
    var domainQuery = new GlideRecord('incident');
    domainQuery.addQuery('active', true);
    domainQuery.addDomainQuery(domain);
    domainQuery.query();
    gs.print('Number of Incidents for ITIL user: ' +
domainQuery.getRowCount());
    while (domainQuery.next())
        gs.print(domainQuery.number);
}
```

addInactiveQuery

public QueryCondition addInactiveQuery()

Adds a filter to return inactive records.

Returns:

`QueryCondition` - records where the active flag is false.

Example:

```
var inc = new GlideRecord('incident');
inc.addInactiveQuery();
inc.query();
```

addJoinQuery

public QueryCondition addJoinQuery(`joinTable, [primaryField], [joinTableField]`)

Adds a filter to return records based on a relationship in a related table.

For example, find all the users that are in the database group (users via *sys_user_grmember* table). Another example would be find all problems that have an assigned incident (problems via the *incident.problem_id* relationship).

Note: this is not a true database join; rather, `addJoinQuery` adds a subquery. So, while the result set is limited based on the join, the only fields that you have access to are those on the base table (those which are in the table with which the GlideRecord was initialized).

Parameters:

`joinTable` - table name.

`primaryField` (optional) - if other than sys_id, the primary field.

`joinTableField` (optional) - if other than sys_id, the field that joins the tables.

Returns:

`QueryCondition` - records where the relationships match.

Example:

Find problems that have an incident attached. This example returns problems that have associated incidents. However, it won't pull values from the incidents that are returned as a part of the query.

```
var prob = new GlideRecord('problem');
prob.addJoinQuery('incident');
prob.query();
```

Example:

Find active=false problems with associated incidents.

```
/ Look for Problem records
var gr = new GlideRecord('problem');

// That have associated Incident records
var grSQ = gr.addJoinQuery('incident');

// Where the Problem records are "active=false"
gr.addQuery('active', 'false');

// And the Incident records are "active=true"
grSQ.addCondition('active', 'true');

// Query
gr.query();

// Iterate and print results
while (gr.next()) {
    gs.print(gr.getValue('number'));
}
```

Example:

Find problems that have incidents associated where the incident **caller_id** field value matches that of the problem **opened_by** field.

```
var gr = new GlideRecord('problem');
gr.addJoinQuery('incident', 'opened_by', 'caller_id');
gr.query();
```

addNotNullQuery

public QueryCondition addNotNullQuery(`String fieldName`)

Adds a filter to return records where the specified field is not null.

Parameters:

`fieldName` - string name for a field.

Returns:

`QueryCondition` - QueryCondition of records where the parameter field is not null.

Example:

```
var target = new GlideRecord('incident');
  target.addNotNullQuery('short_description');
  target.query();    // Issue the query to the database to get all
records
  while (target.next()) {
     // add code here to process the incident record
  }
```

addNullQuery

public QueryCondition addNullQuery(`String fieldName`)

Adds a filter to return records where the specified field is null.

Parameters:

`fieldName` - string name for a field.

Returns:

`QueryCondition` - QueryCondition of records where the parameter field is null.

Example:

```
var target = new GlideRecord('incident');
  target.addNullQuery('short_description');
  target.query();    // Issue the query to the database to get all
records
  while (target.next()) {
     // add code here to process the incident record
  }
```

addOrCondition

public QueryCondition addOrCondition(`String name, String oper, Object value`)

Appends an OR condition to an existing condition based on the parameters entered. When using multiple query parameters, the addOrCondition method adds an OR condition to the last query.

Adds a disjunctive filter condition to return records based on 1, 2 or 3 arguments.

- 1 argument adds an encoded query string.
- 2 arguments return records where the field is equal to the value (or is in a list of values).
- 3 arguments return records where the field meets the specified condition (field, operator and value).

Parameters:

`name` - string name of a field.

`oper` - an operator for the query.

`value` - value to query on.

Returns:

`QueryCondition` - a reference to the QueryCondition that was added to the GlideRecord.

Example:

```
var gr = new GlideRecord('incident');
var qc = gr.addQuery('category', 'hardware');
qc.addOrCondition('category', 'software');
gr.addQuery('priority', '1');
```

addQuery

public QueryCondition addQuery(`String name, Object operator, Object value`)

Adds a filter to return records based on 1, 2 or 3 arguments.

- 1 argument adds an encoded query string.
- 2 arguments return records where the field is equal to the value (or is in a list of values).
- 3 arguments return records where the field meets the specified condition (field, operator and value).

Can accept an `IN` operator (see example 2).

Parameters:

`name` - string name of a field.

`operator` - an operator for the query.

`value` - value to query on.

Returns:

`QueryCondition` - a reference to the QueryCondition that was added to the GlideRecord.

Example 1:

```
var rec = new GlideRecord('incident');
rec.addQuery('active',true);
rec.addQuery('sys_created_on', ">", "2010-01-19 04:05:00");
rec.query();
while (rec.next()) {
 rec.active = false;
 gs.print('Active incident ' + rec.number + ' closed');
 rec.update();
}
```

Example 2 - Using the IN Operator:

```
var que = new GlideRecord('incident');
que.addQuery('number','IN','INC00001,INC00002');
que.query();
while(que.next()) {
 //do something....
}
```

Controlling invalid queries

By default queries with invalid field names run but ignore the invalid condition. For more strict query control you can enable the glide.invalid_query.returns_no_rows property which will result in an empty result set for invalid queries.

When this property is enabled you can temporarily override this logic on the session by turning off strict query control.

```
gs.getSession().setStrictQuery(false);
```

Example:

```
/*
 * Default behavior if no property or property = false
 * Remove invalid query condition
 */
var gr = new GlideRecord("incident");
gr.addQuery("field_that_doesnt_exist", "my value");
gr.addQuery("active", true);
gr.query();
//returns records matching active=true

/*
 * Property = true
 * Invalidate entire query if invalid condition exists
 */
var gr = new GlideRecord("incident");
gr.addQuery("field_that_doesnt_exist", "my value");
gr.addQuery("active", true);
gr.query();
// returns no records
//also generated log message "field name 'field_that_doesnt_exist' not
found in table 'incident'"


/*
 * Property = true, override strict query for session
 * Same behavior as setting property = false except only applies to
this session
 */
//disable strict query
gs.getSession().setStrictQuery(false);

var gr = new GlideRecord("incident");
gr.addQuery("field_that_doesnt_exist", "my value");
gr.addQuery("active", true);
gr.query();
//returns records matching active=true

//restore strict query
gs.getSession().setStrictQuery(true);
```

canCreate

public boolean canCreate()

Determines if the Access Control Rules (which includes the user's role) permit inserting new records in this table.

Returns:

`boolean` - true if the user's role permits creation of new records in this file.

canDelete

public boolean canDelete()

Determines if the Access Control Rules (which includes the user's role) permit deletion of records in this table.

Returns:

`boolean` - true if can delete or false if cannot delete.

Example:

Note: *This API call changed in the Calgary release:*

- *GlideSecurityManager* replaces *Packages.com.glide.sys.security.GlideSecurityManager*

The new script object calls apply to the Calgary release and beyond. For releases prior to Calgary, substitute the packages call as described above. Packages calls are not valid beginning with the Calgary release. For more information, see Scripting API Changes.

```
var att = new GlideRecord('sys_attachment');
att.get('$[sys_attachment.sys_id]');
var sm = GlideSecurityManager.get();
var checkMe = 'record/sys_attachment/delete';
var canDelete = sm.hasRightsTo(checkMe, att);
gs.log('canDelete: ' + canDelete);
```

canRead

public boolean canRead()

Determines if the Access Control Rules (which includes the user's role) permit reading this table.

Returns:

`boolean` - true if can read or false if cannot read.

canWrite

public boolean canWrite()

Determines if the Access Control Rules (which includes the user's role) permit updates to records in this table.

Returns:

`boolean` - true if can write or false if cannot write to the table.

changes

public boolean changes()

Determines whether any of the fields in the record have changed.

Returns:

`boolean` - true if any of the fields in the record have changed.

find

public boolean find(`columnName, value`)

Returns true if any record has a matching value in the specified column. If found, it also moves to the first record that matches, essentially executing next() until the record is returned.

Parameters:

`columnName` - specifies the column name in a record.

`value` - specifies the value to check for in the specified column.

Returns:

`boolean` - true if any record has a matching value in the specified column.

hasAttachments

public boolean hasAttachments()

Determines if the current record has any attachments.

Returns:

`boolean` - true if the current record has attachments.

Example:

```
  //Check for attachments and add link if there are any
  var attachment_link = '';
  var rec = new GlideRecord('sc_req_item');
  rec.addQuery('sys_id', current.request_item);
  rec.query();
  if(rec.next()){
    if(rec.hasAttachments()){
      attachment_link = gs.getProperty('glide.servlet.uri') +
rec.getLink();
    }
  }
```

hasNext

public boolean hasNext()

Determines if there are any more records in the GlideRecord.

Parameters:

`boolean` - true if there are more records in the query set.

Example:

```
if (gr.hasNext()) {
  dothis(); // found it, do it
} else {
  dothat(); // didn't find it
}
;
```

instanceOf

public boolean instanceOf(`String className`)

Checks a table for a type/class of record.

Parameters:

`className` - A string name of a type or class of record.

Returns:

`boolean` - true if table is instance of specified class.

Example:

isNewRecord

public boolean isNewRecord()

Determines whether the current record has been inserted into the database. This method returns true only if the `newRecord()` method has been called. This method is useful for scripted ACL, and in the condition of UI actions, but should not be used in background scripts.

Note: This method returns true for any new record during a business rule, or if the `newRecord()` method is used to initialize a record with default values and a unique ID (sys_id). In all other cases, it returns false.

Returns:

`boolean` - true if the current record is new (not inserted into the database).

Example:

Example of scripted ACL:

```
answer = gs.hasRole("filter_admin") || current.isNewRecord()
```

Example UI action condition:

```
*** Script: true
*** Script: false
```

isValid

public boolean isValid()

Determines whether the table exists or not.

Returns:

`boolean` - true if table is valid or if record was successfully fetched, or false if table is invalid or record was not successfully fetched.

Example:

```
var testTable = new GlideRecord('incident');
gs.print(testTable.isValid());

var testTable = new GlideRecord('wrong_table_name');
gs.print(testTable.isValid());
```

Output:

```
*** Script: true
*** Script: false
```

isValidField

public boolean isValidField(`String columnName`)

Determines if the given field is defined in the current table.

Parameters:

`columnName` - column name of a field as a string.

Returns:

`boolean` - true if field is valid.

isValidRecord

public boolean isValidRecord()

Determine if there is another record in the GlideRecord.

Returns:

`boolean` - true if the current record is valid or false if past the end of the recordset.

next

public boolean next()

Moves to the next record in the GlideRecord.

Returns:

`boolean` - false if there are no more records in the query set .

Example:

```
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
  gs.print(rec.number + ' exists');
}
```

_next

public boolean _next()

Provides the same functionality as `next`, intended to be used in cases where the GlideRecord has a column named **next.**

Moves to the next record in the GlideRecord.

Returns:

`boolean` - false if there are no more records in the query set .

Example:

```
var rec = new GlideRecord('sys_template');
rec.query();
while (rec._next()) {
  gs.print(rec.number + ' exists');
}
```

---

operation

public String operation()

Retrieves the current operation being performed, such as insert, update, delete, etc.

Returns:

`String` - the current operation.

---

orderBy

public void orderBy(`String name`)

Specifies an orderBy column (this may be called more than once to order by multiple columns).

Parameters:

`name` - a column name used to order the recordset.

Example:

```
function UpdateProjectWBS(project) {
  var count = 0;
  var child = new GlideRecord('pm_project_task');
  child.addQuery('parent', project.sys_id);
  child.orderBy('order');
  child.orderBy('number');
  child.query();
  var len = child.getRowCount().toString().length;
  var seq = 0;
  while (child.next()) {
    count += UpdateProjectTaskWBS(child, 1, ++seq, len, '');
  }
  gs.addInfoMessage(count + ' Project Tasks updated');
}
```

orderByDesc

public void orderByDesc(`String name`)

Specifies a decending orderBy column.

Parameters:

`name` - a column name used to order the GlideRecord.

query

public void query(`Object field, Object value`)

Runs the query against the table based on the filters specified by addQuery and addEncodedQuery. This will query the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but a name/value pair can be specified.

Parameters:

`name` - a column name to query on.

`value` - a value to query for.

Example:

```
var rec = new GlideRecord('incident');
rec.query();
while (rec.next()) {
 gs.print(rec.number + ' exists');
}
```

queryNoDomain

public void query(`Object field, Object value`)

Used in domain separated instances. Similar to #query, runs the query against the table based on the filters specified by addQuery and addEncodedQuery, but ignores domains. This will query the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but a name/value pair can be specified.

Parameters:

`name` - a column name to query on.

`value` - a value to query for.

Example:

```
var rec = new GlideRecord('incident');
rec.queryNoDomain();
while (rec.next()) {
 gs.print(rec.number + ' exists');
}
```

_query

public void _query(`Object field, Object value`)

Identical to `query();`, intended to be used on tables where there is a column named **'query**, which would interfere with using query();.

Runs the query against the table based on the filters specified by addQuery and addEncodedQuery. This will query the GlideRecord table as well as any references of the table. One argument adds a query string. Usually this is performed without arguments, but a name/value pair can be specified.

Parameters:

`name` - a column name to query on.

`value` - a value to query for.

Example:

```
var rec = new GlideRecord('sys_app_module');
rec._query();
while (rec.next()) {
 gs.print(rec.number + ' exists');
}
```

restoreLocation

public void restoreLocation(`boolean b`)

Set the current record to be the record that was saved with saveLocation(). If saveLocation has not been called yet, the current record is set to be the first record of the GlideRecord.

saveLocation

public void saveLocation()

Save the current row number so that we can get back to this location using restoreLocation().

Get Methods

get

public boolean get(`Object name, Object value`)

Defines a GlideRecord based on the specified expression of 'name = value'. If value is not specified, then the expression used is 'sys_id = name'. This method is expected to be used to query for single records, so a 'next' operation on the recordset is performed by this method before returning.

Parameters:

`name` - column name.

`value` - (optional) value to match.

Returns:

`name` - true if a matching record(s) was found or false if no matches found.

Example:

```
function kbWriteComment(id, comments) {
 var fb = new GlideRecord('kb_feedback');
 fb.initialize();
 fb.article = id;
 fb.comments = comments;
 fb.insert();
}

function kbGetText(id) {
 var gr = new GlideRecord('kb_knowledge');
 if (gr.get(id)) {
 gr.u_times_used = gr.u_times_used + 1;
 gr.u_last_used = gs.nowDateTime();
 gr.update();
 // return "Knowledge article " + gr.number + ":\n" +
 // "[code]" + gr.text + "[/code]";
 return gr.number;
 }
```

getAttribute

Gets the attributes on the field in question from the dictionary.

public String getAttribute(`String`)

Retrieves the attributes on the field in question from the dictionary.

Parameters:

`String` - the column name as a string to get the attributes from.

Returns:

`String` - the attribute values as a string.

Example:

```
doit();
function doit() {
  var gr = new GlideRecord('sys_user');
  gr.query("user_name","admin");
  if (gr.next()) {
    gs.print("we got one");
    gs.print(gr.location.getAttribute("tree_picker"));
  }
}
```

getClassDisplayValue

public String getClassDisplayValue()

Retrieves the label for the table.

Returns:

`String` - label that describes the table.

getDisplayValue

public String getDisplayValue()

Retrieves the display value for the current record.

Returns:

`String` - a string display value for the current record.

Example:

```
// list will contain a series of display values separated by a comma
  // array will be a javascript array of display values
  var list = current.watch_list.getDisplayValue();
  var array = list.split(",");
  for (var i=0; i < array.length; i++) {
     gs.print("Display value is: " + array[i]);
  }
```

getED

public ElementDescriptor getED()

Retrieves the element's descriptor.

Returns:

`ElementDescriptor` - the current element's descriptor.

Example:

```
var totalCritical  = 0;
var filledCritical = 0;
var fields         = current.getFields();

gs.print(fields);

for (var num = 0; num < fields.size(); num++) {

      gs.print("RUNNING ARRAY VALUE " + num);

      var ed = fields.get(num).getED();

      if(ed.hasAttribute("tiaa_critical")) {
            gs.print("CRITICAL FIELD FOUND");
            totalCritical ++;

            if (!fields.get(num).isNil()) {
                  filledCritical ++;
            }
      }

}

var answer = 0;
gs.print("TOTAL - " + totalCritical);
gs.print("FILLED - " + filledCritical);

if (filledCritical > 0 && totalCritical > 0) {

      var pcnt = (filledCritical/totalCritical)*100;
      answer = pcnt.toFixed(2);;

}

answer;
```

getElement

public GlideElement getElement(`String columnName`)

Retrieves the GlideElement for a specified field.

Parameters:

`columnName` - a column name to get the element from.

Returns:

`GlideElement` - a GlideElement.

getEncodedQuery

public String getEncodedQuery(`boolean b`)

Retrieves the encoded query as a string.

Parameters:

`String` - Gets the encoded query as a string.

getEscapedDisplayValue

public String getEscapedDisplayValue()

Retrieves the field value for the display field of the current record and escape it for use in Jelly scripts.

Returns:

`String` - escaped value of display column.

getFields

public ArrayList getFields()

Retrieves a Java ArrayList of fields in the current record.

Parameters:

`ArrayList` - a Java ArrayList of fields in the current record.

Example:

```
// This can be run in "Scripts - Background" for demonstration purposes

// Get a single incident record
var grINC = new GlideRecord('incident');
grINC.query();
grINC.next();
gs.print('Using ' + grINC.getValue('number'));
gs.print('');

// getFields() returns a Java ArrayList
var fields = grINC.getFields();

// Enumerate GlideElements in the GlideRecord object that have values
gs.print('Enumerating over all fields with values:');
for (var i = 0; i < fields.size(); i++) {
  var glideElement = fields.get(i);
  if (glideElement.hasValue()) {
    gs.print('  ' + glideElement.getName() + '\t' + glideElement);
  }
}
gs.print('');

// Get a specific GlideElement: number
gs.print('Getting the number field:');
for (var i = 0; i < fields.size(); i++) {
  var glideElement = fields.get(i);
  if (glideElement.hasValue() && glideElement.getName() == 'number') {
    gs.print('  ' + glideElement.getName() + '\t' + glideElement);
  }
}
```

getLabel

public String getLabel()

Retrieves the appropriate label for a field.

Returns:

`String` - the label of the field as a string.

Example:

```
template.print("Summary of Requested items:\n");
var gr = new GlideRecord("sc_req_item");
gr.addQuery("request", current.sysapproval);
gr.query();
while(gr.next()) {
    var nicePrice = gr.price.toString();
    if (nicePrice != '') {
        nicePrice = parseFloat(nicePrice);
        nicePrice = nicePrice.toFixed(2);
    }
    template.print(gr.number + ":  " + gr.quantity + " X " +
gr.cat_item.getDisplayValue() + " at $" + nicePrice + " each \n");
    template.print("    Options:\n");
    for (key in gr.variables) {
      var v = gr.variables[key];
      if(v.getGlideObject().getQuestion().getLabel() != '') {
         template.space(4);
         template.print('     ' +
v.getGlideObject().getQuestion().getLabel() + " = " +
v.getDisplayValue() + "\n");
      }
    }
}
```

getLink

public String getLink(`boolean noStack`)

Retrieves the link for the current record.

Parameters:

`noStack` - if true, the link generated will not append **&sysparm_stack=[tablename]_list.do?sysparm_query=active=true** to the end of the URL; if false, the link will. Leaving the argument empty will default to false.

Returns:

`String` - a URL.

Example:

```
  //Check for attachments and add link if there are any
  var attachment_link = '';
  var rec = new GlideRecord('sc_req_item');
  rec.addQuery('sys_id', current.request_item);
  rec.query();
  if(rec.next()){
    if(rec.hasAttachments()){
      attachment_link = gs.getProperty('glide.servlet.uri') +
rec.getLink();
    }
  }
```

getLocation

public int getLocation(`boolean b`)

Retrieves the current row number.

Returns:

`int` - Returns the row number of the current record.

getPlural

public string getPlural()

Retrieves the plural label of the GlideRecord table.

Returns:

`string` - the plural label of the GlideRecord's table.

For example, if the table name is "Change Request," this method would return "Change Requests."

getRecordClassName

public String getRecordClassName()

Retrieves the class name for the current record.

Parameters:

`String` - class or table name.

Example:

```
function TaskAssignmentFilter() {
  var classname = current.getRecordClassName();
  var filter = "type=null";
  if (classname == "incident" && current.category == "database") {
    filter = GetGroupFilter("database");
  }
  else {
    // append exclusion for 'catalog' to the filter
    var cat = new GlideRecord("sys_user_group_type");
    cat.addQuery("name", "catalog");
    cat.query();
    if (cat.next()) {
      filter += "^ORtype!=" + cat.sys_id;
    }
  }
  gs.log("TaskAssignmentFilter: " + filter);
  return filter;
}
```

getRelatedLists

public HashMap getRelatedLists(`boolean b`)

Retrieves a list of names and display values of tables that **refer to** the current record.

Returns:

`HashMap` - hashmap with names and display values of related tables.

getRelatedTables

public HashMap getRelatedTables(`boolean b`)

Retrieves a list of names and display values of tables that are**referred to by** the current record.

Parameters:

`HashMap` - hashmap with names and display values of related tables.

getRowCount

public int getRowCount()

Retrieves the number of rows in the GlideRecord.

Returns:

`int` - an integer count of rows.

Example:

```
var gr = new GlideRecord('incident')
gr.query();
gs.info("Records in incident table: " + gr.getRowCount());
```

getRowNumber

public int getRowNumber()

Retrieves the row number set by saveLocation() or setLocation(). To get the current row, use getLocation() instead.

Returns:

`int` - the saved row number.

getTableName

public String getTableName()

Retrieves the table name associated with this GlideRecord.

Returns:

`String` - table name.

Example:

```
gs.log('Table: ' + current.getTableName());
gs.log('Parent: ' + current.parent.sys_id);
var item = new GlideRecord('sc_req_item');
item.addQuery('sys_id', current.parent.sys_id);
item.query();
if(item.next()){
  for(var variable in item.variable_pool) { gs.log(variable);
  var answer = eval ("item.variable_pool." + variable +
".getDisplayValue()");
gs.log(answer);}
}
```

getValue

public String getValue(`String fieldName`)

Retrieves the string value of an underlying element in a field.

Parameters:

`fieldName` - the name of the field from which to get the value.

Returns:

`String` - the string value of the underlying element. Returns null if the field is empty, or the field does not exist.

Set Methods

autoSysFields

public void autoSysFields(`boolean e`)

Note: *This is not available for scoped apps, starting with the Fuji release. See the Scoped GlideRecord API Reference for a list of what APIs are available for scoped apps.*

Enables or disables the update to the fields sys_updated_by, sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on. This is often used for manually updating field values on a record while leaving historical information unchanged.

Note: Use caution if you use this method. When you use this method the **sys_mod_count** field will not be incremented, and other **sys_** fields will not be updated. This can break functionality including, but not limited to, the Activity Formatter, History Sets, and Metrics.

Parameters:

`e` - Boolean variable that if false disables updates to sys_updated_by, sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on.

Example:

```
var inc = new GlideRecord('incident');

// Change all Open(1) incidents to Active(2)

inc.addQuery('state', 1);
inc.query();

while (inc.next()) {
  inc.autoSysFields(false);  // Do not update sys_updated_by,
sys_updated_on, sys_mod_count, sys_created_by, and sys_created_on
  inc.setWorkflow(false);    // Do not run any other business rules
  inc.setValue('state', 2);
  inc.update();
}
```

setAbortAction

public void setAbortAction(`boolean b`)

Sets a flag to indicate if the next database action (insert, update, delete) is to be aborted.

Parameters:

`b` - true to abort next action or false if action is to be allowed.

Example:

```
if ((!current.u_date1.nil()) && (!current.u_date2.nil())) {
  var start = current.u_date1.getGlideObject().getNumericValue();
  var end = current.u_date2.getGlideObject().getNumericValue();
  if (start > end) {
    gs.addInfoMessage('start must be before end');
    current.u_date1.setError('start must be before end');
    current.setAbortAction(true);
  }
}
```

setDisplayValue

public void setDisplayValue(`String name, Object value`)

Sets the field name to the specified display value.

- For a reference field this is the display value for the table.
- For a date/time this is the time in the caller's current timezone.

Parameters:

`name` - String value of the field name.

`value` - Display value to set for the field.

Example:

```
var gr = new GlideRecord('incident');
gr.get('46f09e75a9fe198100f4ffd8d366d17b');
gr.setDisplayValue('opened_at','2011-02-13 4:30:00');
gr.update();
```

setForceUpdate

public void setForceUpdate(`boolean force`)

Updates the record even if fields have not been changed.

Parameters:

`force` - true to update even if fields have not changed, otherwise false.

setLimit

public void setLimit(`int`)

Sets the limit for how many records are in the GlideRecord.

Parameters:

`int` - Integer limit for records to fetch.

Example:

```
var gr = new GlideRecord('incident');
gr.orderByDesc('sys_created_on');
gr.setLimit(10);
gr.query();
```

setLocation

public void setLocation(`int rowNumber`)

Sets the current row location.

Parameters:

`rowNumber` - integer that is the number of the row to set as the current row.

setNewGuid

public void setNewGuid()

Generates a new GUID and sets it as the unique id for the current record. This function applies only to new records. The GUID for an existing record cannot be changed.

Example:

```
var tsk_id = task.setNewGuid();

task.description = "Request: " + current.request.number;
task.description = task.description + "\n" + "Requested by: " +
current.request.u_requested_by.name;
task.description = task.description + "\n" + "Requested for: " +
current.request.u_requested_for.name;
task.description = task.description + "\n" + "Item: " +
current.cat_item.name;

var gr = new GlideRecord ('task_rel_task');
//link the incident to the request (may need to review if it needs to
be the item)
gr.parent = current.request;
gr.child = tsk_id;
gr.insert();
```

setNewGuidValue

public void setNewGuidValue(`String guid`)

Generates a new GUID and sets it as the unique id for the current record, when inserting a new record.

Parameters:

`guid` - a string value for the new GUID.

setQueryReferences

public void setQueryReferences(`boolean queryReferences`)

Enables or disables using the reference field's display name when querying a reference field.

Parameters:

`queryReferences` - Boolean variable. If set to true, will generate a string of display names; if false, will generate a string of sys_ids.

setUseEngines

public void setUseEngines(`boolean e`)

Disable or enable the running of any engines (approval rules / assignment rules).

Parameters:

`e` - if true enables engines, and if false disables engines.

setValue

public void setValue(`String name, Object value`)

Sets the specified field to the specified value. Normally a script would do a direct assignment, for example, gr.category = value. However, if in a script the element name is a variable, then `gr.setValue(elementName, value)` can be used. When setting a value, ensure the data type of the field matches the data type of the value you enter. This method cannot be used on journal fields.

If the value parameter is null, the record is not updated, and an error is not thrown.

Parameters:

`name` - a field name.

`value` - an object value.

setWorkflow

public void setWorkflow(`boolean e`)

Enables or disables the running of business rules that might normally be triggered by subsequent actions. If the `e` parameter is set to false, an insert/update will not be audited. Auditing only happens when the parameter is set to true for a GlideRecord operation.

Parameters:

`e` - Boolean variable that if true (default) enables business rules, and if false to disables them.

Example:

```
doit('name1','name2');

function doit(username1,username2) {

  var usr1 = new GlideRecord('sys_user');
  var usr2 = new GlideRecord('sys_user');
  var num = 0;

  if (usr1.get('user_name',username1) &&
usr2.get('user_name',username2)) {
    var ref;
    var dict = new GlideRecord('sys_dictionary');
    dict.addQuery('reference','sys_user');
    dict.addQuery('internal_type','reference');
    dict.query();
    while (dict.next()) {
      num = 0;
      ref = new GlideRecord(dict.name.toString());
      ref.addQuery(dict.element,usr1.sys_id);
      ref.query();
      while (ref.next()) {
        ref.setValue(dict.element.toString(),usr2.sys_id);
        ref.setWorkflow(false);
        ref.update();
        num++;
      }
      if (num > 0) {
        gs.print(dict.element + ' changed from ' + usr1.user_name +
          ' to ' + usr2.user_name + ' in ' + num + ' ' + dict.name + '
records');
      }
    }
  }
}
```

Note:setWorkflow() is ignored when subsequently using either deleteProblem() or deleteMultiple() to cascade delete.

Update Methods

applyTemplate

public void applyTemplate(`String template`)

Apply a template record (from **sys_template**) to the current record. If the specified template is not found, no action is taken.

Parameters:

`template` - Takes a name of a template on the **sys_template** table.

Example:

```
var rec1 = new GlideRecord("incident");
rec1.initialize();
rec1.applyTemplate("my_incident_template");

//...possibly more code here...
rec1.insert();
```

update

public string update(`Object reason`)

Updates the GlideRecord with any changes that have been made. If the record does not already exist, it is inserted.

Parameters:

`reason` - Takes a string designating the reason for the update if necessary. The reason will be displayed in the audit record.

Returns:

`String` - unique id of new or updated record.

Example:

```
  var gr = new GlideRecord('task_ci');
gr.addQuery();
gr.query();
var count = gr.getRowCount();
if (count > 0) {
   var allocation = parseInt(10000 / count) / 100;
   while (gr.next()) {
      gr.u_allocation = allocation;
      gr.update();
   }
}
```

updateWithReferences

public string updateWithReferences(`Object reason`)

Updates a record and also inserts or updates any related records with the information provided.

Returns:

`string` - unique ID for the record being updated.

Parameters:

`reason` - takes a string designating the reason for the update if necessary. The reason will be displayed in the audit record.

Example

- if processing a incident where the Caller ID is set to reference sys_user record 'David Loo,' then the following code would update David Loo's user record.
- if processing a incident where there is no Caller ID specified, then the following code would create a new sys_user record with the provided information (first_name, last_name) and set the Caller ID value to the newly created sys_user record.

```
var inc = new GlideRecord('incident');
inc.get(inc_sys_id);  // Looking up an existing incident record where
'inc_sys_id' represents the sys_id of a incident record
inc.caller_id.first_name = 'John';
inc.caller_id.last_name = 'Doe';
inc.updateWithReferences();
}
```

Insert Methods

initialize

Creates an empty record suitable for population before an insert.

public void initialize()

Example:

```
var gr = new GlideRecord('to_do');
gr.initialize();
gr.name = 'first to do item';
gr.description = 'learn about GlideRecord';
gr.insert();
```

insert

public string insert()

Inserts a new record using the field values that have been set for the current record.

Returns:

`string` - unique id of the inserted record or null if record is not inserted.

Example:

```
var gr = new GlideRecord('to_do');
gr.initialize();
gr.name = 'first to do item';
gr.description = 'learn about GlideRecord';
gr.insert();
```

insertWithReferences

public string insertWithReferences()

Inserts a new record and also inserts or updates any related records with the information provided.

Returns:

`string` - unique ID for the record being inserted.

Example

If a reference value is not specified (as below), then a new user record will be created with the provided first_name, last_name, and the caller_id value is set to this newly created sys_user record.

The result is a new sys_user record with the provided first_name, last_name and a new incident record with the provided short_description and caller_id.

```
var inc = new GlideRecord(‘incident’);
inc.initialize();
inc.short_description = 'New incident 1';
inc.caller_id.first_name = 'John';
inc.caller_id.last_name = 'Doe';
inc.insertWithReferences();
}
```

Example

If a caller_id value is specified, then that caller_id is updated with the provided first_name, last_name.

The result is a newly created incident record with values set for short_description and caller_id.

```
var inc = new GlideRecord(‘incident’);
inc.initialize();
inc.short_description = 'New incident 1';
inc.caller_id.setDisplayValue('David Loo');
inc.caller_id.first_name = 'John';
inc.caller_id.last_name = 'Doe';
inc.insertWithReferences();
}
```

newRecord

public void newRecord()

Creates a GlideRecord, set the default values for the fields and assign a unique id to the record. See Also: initialize().

Delete Methods

deleteMultiple

public void deleteMultiple()

Deletes multiple records according to the current "where" clause. Does not delete attachments.

Example:

```
function nukeCart() {
    var cart = getCart();
    var id = cart.sys_id;
    var kids = new GlideRecord('sc_cart_item');
    kids.addQuery('cart', cart.sys_id);
    kids.deleteMultiple();
```

Note: Dot-walking is not supported for this method. When deploying the deleteMultiple() function on referenced tables, all the records in the table will be deleted. Also, when using deleteRecord() to cascade delete, prior calls to setWorkflow() on the same GlideRecord object are ignored.

deleteRecord

public boolean deleteRecord()

Deletes a single record.

Parameters:

`boolean` - true if record was deleted or false if none found to delete.

Example:

```
var gr = new GlideRecord('incident')
gr.addQuery('sys_id','99ebb4156fa831005be8883e6b3ee4b9'); //to delete
one record
gr.query();
gr.next();
gr.deleteRecord();
}
```

Note: When using deleteMultiple() to cascade delete, prior calls to setWorkflow() on the same GlideRecord object are ignored.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/glide-server-apis/concept/c_GlideRecord.html

GlideSystem

Note: *This article applies to Fuji and earlier releases. For more current information, see GlideSystem* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

The GlideSystem (referred to by the variable name 'gs' in Business Rules) provides a number of convenient methods to get information about the system, the current logged in user, etc. For example, the method addInfoMessage() permits communication with the user.

```
gs.addInfoMessage('Email address added for notification');
```

Many of the GlideSystem methods facilitate the easy inclusion of dates in query ranges and are most often used in filters and reporting.

Note: *The global Glide APIs are not available in scoped scripts. There are scoped Glide APIs for use in scoped scripts. The scoped Glide APIs provide a subset of the global Glide API methods. For information on scoped applications see Application Scope, scripting in scoped applications, and the list of scoped APIs.*

The Method Detail is in three sections:

- General functions
- Date and Time functions
- User Session functions

Method Summary

General

Method Summary	Description
eventQueue(String, Object, String, String, String)	Queues an event for the event manager.
getCurrentScopeName()	Gets the name of the current scope.
getDisplayColumn(String)	Gets the display column for the table.
getDisplayValueFor(String, String, String)	Gets the display value for a given field.
getEscapedProperty(String, Object)	Gets the property and escapes it for XML parsing.
getMessage(String, Object)	Retrieves translated messages to display in the UI. If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.

getMessageS(String, Object)	Retrieves translated messages to display in the UI and escapes all ticks ('). If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.
getNodeValue(object, Integer)	Gets the node value for specified index.
getNodeName(Object, Integer)	Returns the node name for specified index.
getProperty(String, Object)	Gets the value of a Glide property.
getScriptError(String)	Returns the script error found in the specified script, if there is one. The script is not executed by this function, only checked for syntax errors.
getStyle(String, String, String)	Returns the style defined for the table, field and value.
getXMLText (String, String)	Gets the xml text for the first node in the xml string that matches the path query.
getXMLNodeList(String)	Constructs an Array of all the nodes and values in an XML document.
log(String message, String source)	Logs a message to the system log and saves it to the syslog table.
logError(String message, String source)	Logs an error to the system log and saves it to the syslog table.
logWarning(String message, String source)	Logs a warning to the system log and saves it to the syslog table.
nil(Object)	Queries an object and returns **true** if the object is null or contains an empty string.
print(String)	Writes a message to the system log. This method does not write the message to the syslog table unless debug has been activated.
tableExists(String)	Determines if a database table exists.
workflowFlush(Object)	Deletes all existing workflow operations for the specified GlideRecord.

Date and Time Functions

Method Summary	Description
beginningOfLastMonth()	Gets the date and time for the beginning of last month in GMT.
beginningOfLastWeek()	Gets the date and time for the beginning of last week in GMT.
beginningOfNextWeek()	Gets the date and time for the beginning of next week in GMT.
beginningOfNextMonth()	Gets the date and time for the beginning of next month in GMT.
beginningOfNextYear()	Gets the date and time for the beginning of next year in GMT.
beginningOfThisMonth()	Gets the date and time for the beginning of this month in GMT.
beginningOfThisQuarter()	Gets the date and time for the beginning of this quarter in GMT.
beginningOfThisWeek()	Gets the date and time for the beginning of this week in GMT.
beginningOfThisYear()	Gets the date and time for the beginning of this week in GMT.
beginningOfToday()	Gets the date and time for the beginning of today in GMT.
beginningOfYesterday()	Gets the date and time for the beginning of yesterday in GMT.
calDateDiff(String, String, boolean)	Calculate the difference between two dates using the default calendar. **Note:** Calendars are now legacy. If Schedules are being used, see Calculate Duration Given a Schedule.
dateDiff(String, String, boolean)	Calculates the difference between two dates. The parameters must be in the user/system date time format.
dateGenerate(String, String)	Generates a date and time for the specified date in GMT.
daysAgo(int)	Gets a date and time for a certain number of days ago. The result is expressed in GMT.
daysAgoEnd(int)	Gets a date and time for end of the day a certain number of days ago.The result is expressed in GMT.

daysAgoStart(int)	Gets a date and time for beginning of the day a certain number of days ago. The result is expressed in GMT.
endOfLastMonth()	Gets the date and time for the end of last month in GMT.
endOfLastWeek()	Gets the date and time for the end of last week in GMT, in the format **yyyy-mm-dd hh:mm:ss.**
endOfLastYear()	Gets the date and time for the end of last year in GMT.
endOfNextMonth()	Gets the date and time for the end of next month in GMT.
endOfNextWeek()	Gets the date and time for the end of next week in GMT.
endOfNextYear()	Gets the date and time for the end of next year in GMT.
endOfThisMonth()	Gets the date and time for the end of this month in GMT.
endOfThisQuarter()	Gets the date and time for the end of this quarter in GMT.
endOfThisWeek()	Gets the date and time for the beginning of this week in GMT.
endOfThisYear()	Gets the date and time for the end of this year in GMT.
endOfToday()	Gets the date and time for the end of today in GMT.
endOfYesterday()	Gets the date and time for the end of yesterday in GMT.
hoursAgo(int)	Gets a date and time for a certain number of hours ago.The result is expressed in GMT.
hoursAgoEnd(int)	Gets a date and time for the end of the hour a certain number of hours ago.The result is expressed in GMT.
hoursAgoStart(int)	Gets a date and time for the start of the hour a certain number of hours ago.The result is expressed in GMT.
lastWeek()	Date and time one week ago in GMT.
minutesAgo(int)	Gets a date and time for a certain number of minutes ago.The result is expressed in GMT.
minutesAgoEnd(int)	Gets a date and time for the end of the minute a certain number of minutes ago.The result is expressed in GMT.
minutesAgoStart(int)	Gets a date and time for a certain number of minutes ago. The result is expressed in GMT.
monthsAgo(int)	Gets a date and time for a certain number of months ago.The result is expressed in GMT.
monthsAgoEnd(int)	Gets a date and time for the last day of the month a certain number of months ago.The result is expressed in GMT.
monthsAgoStart(int)	Gets a date and time for the first day of the month a certain number of months ago.The result is expressed in GMT.
now()	Gets the current date using GMT date time.
nowNoTZ()	Gets the current GMT date time.
nowDateTime()	Gets the current date and time in the user's time zone.
quartersAgo(int)	Gets a date and time for a certain number of quarters ago. The result is expressed in GMT.
quartersAgoEnd(int)	Gets a date and time for the last day of the quarter a certain number of quarters ago. The result is expressed in GMT.
quartersAgoStart(int)	Gets a date and time for the first day of the quarter a certain number of quarters ago. The result is expressed in GMT.
yearsAgo(int)	Gets a date and time for a certain number of years ago.The result is expressed in GMT.
yesterday()	Gets yesterday's time. The result is expressed in GMT.
isFirstDayOfMonth(Object)	Checks whether the date is the first day of the month.
isFirstDayOfWeek(Object)	Checks whether the date is the first day of the week. This uses the ISO standard of Monday being the first day of the week.
isFirstDayOfYear(Object)	Checks whether the date is the first day of the year
isLastDayOfMonth(Object)	Checks whether the date is the last day of the month.
isLastDayOfWeek(Object)	Checks whether the date is the last day of the week.

isLastDayOfYear(Object)	Checks whether the date is the last day of the year.

User Session Functions

Method Summary	Description
addErrorMessage(Object)	Adds an error message for the current session. Session error messages are shown at the top of the form. Use getErrorMessages() to retrieve the list of error messages that are being shown.
addInfoMessage(Object)	Adds an info message for the current session. Session info messages are shown at the top of the form below any error messages. Use getInfoMessages() to retrieve the list of info messages being shown.
addMessage(String, Object)	Adds a message for the current session. Can be called using getMessages(String).
flushMessages()	Clears session messages saved using addErrorMessage(Object) or addInfoMessage(Object). Session messages are shown at the top of the form. In client side scripts use g_form.clearMessages() to clear all session messages.
getErrorMessages()	Gets the list of error messages for the session that were added by addErrorMessage(Object).
getImpersonatingUserDisplayName()	Returns the display name of the impersonating user.
getImpersonatingUserName()	Returns the name of the impersonating user or null if not impersonating.
getInfoMessages()	Gets the list of info messages for the session that were added via addInfoMessage(Object.
getMessages(String)	Gets the list of messages of the specified type for the session that were added via addMessage(String, Object).
getPreference(String, Object)	Gets a user preference.
getSession()	Returns a GlideSession object.
getSessionID()	Accesses the GlideSession Session ID.
getTrivialMessages()	Gets the list of error messages for the session that were added with the trivial flag.
getUser()	Returns a reference to the User object for the current user. More information is available here.
getUserDisplayName()	Returns the name field of the current user (e.g. John Smith, as opposed to smith).
getUserID()	Returns the sys_id of the current user.
getUserName()	Returns the username of the current user (for example, jsmith).
getUserNameByUserID(String)	Gets the username based on a user ID.
hasRole(String)	Determines if the current user has the specified role.
hasRoleInGroup(Object, Object)	Determines if the current user has the specified role within a specified group.
isInteractive()	Checks if the current session is interactive.
isLoggedIn()	Determines if the current user is currently logged in.
setRedirect(Object)	Sets the redirect URI for this transaction. This determines the next page the user will see.
setReturn(Object)	Sets the return URI for this transaction. This determines what page the user will be directed to when they return from the next form.
userID()	Returns the sys_id of the user associated with this session. A shortcut for the more proper getUserID().

General

eventQueue(String, Object, String, String, String)

Queues an event for the event manager.

Input Fields

Parameters:

- Name of the event being queued.
- A GlideRecord object, such as "current".
- An optional parameter, saved with the instance if specified.
- A second optional parameter, saved with the instance if specified.
- An event queue to add the event to.

Example

```
if (current.operation() != 'insert' && current.comments.changes()) {
    gs.eventQueue('incident.commented', current, gs.getUserID(),
gs.getUserName());
}
```

getCurrentScopeName()

Gets the name of the current scope.

Input Fields

Parameters:

- None

Example

```
gs.getCurrentScopeName();
```

getDisplayColumn(String)

Gets the display column for the table.

Input Fields

Parameters: name of table from which to get the display column name.

Output Fields

Returns: name of the display column for the table.

Example

```
// Return the sys_id value for a given table and its display value
function GetIDValue(table, displayValue) {
    var rec = new GlideRecord(table);
    var dn = gs.getDisplayColumn(table);
    if (rec.get(dn, displayValue))
        return rec.sys_id;
    else
        return null;
}
```

getDisplayValueFor(String, String, String)

Gets the display value for a specified field on a record.

Input Fields

Parameters:

- String - name of table for the record
- String - the sysid for the record to get the display value from
- String - the field to get a display value from

Output Fields

Returns: name of the display value for the field's value.

getEscapedProperty(String, Object)

Gets the property and escapes it for XML parsing.

Input Fields

Parameters:

- String key for the property whose value should be returned
- Alternate object to return if the property is not found.

Output Fields

Returns: the property as a string, or the alternate object specified above.

getMessage(String, Object)

Retrieves translated messages to display in the UI. If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned.

Note: if the UI message has a tick ('), there may be issues with the message in the script; to escape the ticks ('), use getMessageS(String, Object).

Input Fields

Parameters:

- ID of message
- (Optional) A list of strings or other values defined by java.text.MessageFormat [2], which allows you to produce language-neutral messages for display to users.

Output Fields

Returns: the UI message as a string.

Examples

```
var my_message = '${gs.getMessage("This is a message.")}';
alert(my_message);
```

Using the optional parameter as in:

```
gs.getMessage('"{0}" is not a Client Callable Script Include','BAR');
```

Returns: "BAR" is not a Client Callable Script Include

getMessageS(String, Object)

Retrieves translated messages to display in the UI and escapes all ticks ('). If the specified string exists in the database for the current language, then the translated message is returned. If the specified string does not exist for the current language, then the English version of the string is returned. If the string does not exist at all in the database, then the ID itself is returned. Useful if you are inserting into a JavaScript expression from Jelly.

For instance, the following code snippet will fail if the returned snippet has a tick ('):

```
var my_message = '${gs.getMessage("I love France")}';
alert(my_message);
```

The solution is to use a snippet like the following:

```
var my_message = '${gs.getMessageS("I love France")}';
alert(my_message);
```

Input Fields

Parameters:

- ID of message
- Optional message arguments.

Output Fields

Returns: the message as a string, with the ticks escaped.

Example

```
var my_message = '${gs.getMessageS("I love France")}';
alert(my_message);
```

getProperty(String, Object)

Gets the value of a Glide property. If the property is not found, return an alternate value. Use getEscapedProperty(String, Object) to escape the property.

Input Fields

Parameters:

- String key for the property whose value should be returned.
- Alternate object to return if the property is not found.

Output Fields

Returns: (String) the value of the Glide property, or the alternate object defined above.

Example

```
  //Check for attachments and add link if there are any
  var attachment_link = '';
  var rec = new GlideRecord('sc_req_item');
  rec.addQuery('sys_id', current.request_item);
  rec.query();
  if(rec.next()){
    if(rec.hasAttachments()){
      attachment_link = gs.getProperty('glide.servlet.uri') +
rec.getLink();
    }
  }
```

getScriptError(String)

Returns the script error found in the specified script, if there is one. The script is not executed by this function, only checked for syntax errors.

Input Fields

Parameters: string script to check for errors.

Output Fields

Returns: the script error message or, if none, then **null.**

getStyle(String, String, String)

Returns the style defined for the table, field and value.

Input Fields

Parameters:

- Table name
- Field name
- Field value

Output Fields

Returns the style as a string.

log(String message, String source)

Logs a message to the system log and saves it to the syslog table.

Input Fields

Parameters:

- **String message** - message to log, for the log's **Message** field.
- **String source** - (optional) the source of the message, for the log's **Source** field.

Example

```
var count = new GlideAggregate('incident');
count.addQuery('active', 'true');
count.addAggregate('COUNT', 'category');
count.query();
while (count.next()) {
   var category = count.category;
   var categoryCount = count.getAggregate('COUNT', 'category');
   gs.log("The are currently " + categoryCount + " incidents with a
category of " + category, "Incident Counter");
}
```

logError(String message, String source)

Logs an error to the system log and saves it to the syslog table.

Input Fields

Parameters:

- **String message** - message to log, for the log's **Message** field.
- **String source** - (optional) the source of the message, for the log's **Source** field.

logWarning(String message, String source)

Logs a warning to the system log and saves it to the syslog table.

Input Fields

Parameters:

- **String message** - message to log, for the log's **Message** field.
- **String source** - (optional) the source of the message, for the log's **Source** field.

nil(Object)

Queries an object and returns **true** if the object is null or contains an empty string.

Input Fields

Parameters: name of an object

Output Fields

Returns: true if null or empty string; otherwise, returns false

Example

```
if ((!current.u_date1.nil()) && (!current.u_date2.nil())) {
  var start = current.u_date1.getGlideObject().getNumericValue();
  var end = current.u_date2.getGlideObject().getNumericValue();
  if (start > end) {
    gs.addInfoMessage('start must be before end');
    current.u_date1.setError('start must be before end');
    current.setAbortAction(true);
  }
}
```

print(String)

Writes a message to the system log. This method does not write the message to the syslog table unless debug has been activated.

Input Fields

Parameters: message to log

Example

```
var rec = new GlideRecord('incident');
rec.addQuery('active',false);
rec.query();
while (rec.next()) {
 gs.print('Inactive incident ' + rec.number + ' deleted');
 rec.deleteRecord();
}
```

tableExists(String)

Determines if a database table exists.

Input Fields

Parameters: name of table to check for existence

Output Fields

Returns: true if table exists or false is not found

workflowFlush(Object)

Deletes all existing workflow operations for the specified GlideRecord.

Input Fields

Parameters: the GlideRecord to flush the workflow for

Output Fields

Returns: void

Date/Time

beginningOfLastMonth()

Gets the date and time for the beginning of last month in GMT.

Output Fields

Returns: the GMT beginning of last month, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfLastWeek()

Gets the date and time for the beginning of last week in GMT.

Output Fields

Returns: the GMT beginning of last week, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfNextWeek()

Gets the date and time for the beginning of next week in GMT.

Output Fields

Returns: the GMT beginning of next week, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfNextMonth()

Gets the date and time for the beginning of next month in GMT.

Output Fields

Returns: the GMT beginning of next month, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfNextYear()

Gets the date and time for the beginning of next year in GMT.

Output Fields

Returns: the GMT beginning of next year, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfThisMonth()

Gets the date and time for the beginning of this month in GMT.

Output Fields

Returns: the GMT beginning of this month, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfThisQuarter()

Gets the date and time for the beginning of this quarter in GMT.

Output Fields

Returns: the GMT beginning of this quarter, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfThisWeek()

Gets the date and time for the beginning of this week in GMT.

Output Fields

Returns: the GMT beginning of this week, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfThisYear()

Gets the date and time for the beginning of this week in GMT.

Output Fields

Returns: the GMT beginning of this week, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfToday()

Gets the date and time for the beginning of today in GMT.

Output Fields

Returns: the GMT beginning of today, in the format **yyyy-mm-dd hh:mm:ss.**

beginningOfYesterday()

Gets the date and time for the beginning of yesterday in GMT.

Output Fields

Returns: the GMT beginning of yesterday, in the format **yyyy-mm-dd hh:mm:ss.**

calDateDiff(String, String, boolean)

Calculate the difference between two dates using the default calendar. **Note:** Calendars are now legacy. If Schedules are being used, see Calculate Duration Given a Schedule.

Input Fields

Parameters:

- **startDate** - a starting date to compare, in the current user's date format
- **endDate** - an ending date to compare, in the current user's date format
- **boolean numericValue** - if true, the return will be formatted in number of seconds; if false, the return will be formatted ddd hh:mm:ss.

Output Fields

Returns: if the numericValue boolean parameter is true, returns the difference between the two dates as an integer number of seconds; if false, returns the difference between the two dates in the format ddd hh:mm:ss.

dateDiff(String, String, boolean)

Calculates the difference between two dates. This method expects the earlier date as the first parameter and the later date as the second parameter; otherwise, the method returns the difference as a negative value. **Note:** Use getDisplayValue() to convert the strings to the expected format.

Input Fields

Parameters:

- **startDate** - a starting date to compare, in the current user's date format.
- **endDate** - an ending date to compare, in the current user's date format.
- **boolean bnumericValue** - true to return difference in number of seconds as a string, false to return difference in the format ddd hh:mm:ss.

Output Fields

Returns: if boolean bnumericValue is true, the difference in number of seconds; if false, the difference in the format ddd hh:mm:ss.

Example

For more examples, see Setting the Duration Field Value.

```
// Given two date/times as DateTime objects
// Set the values this way to ensure a consistent input time
var date1 = new GlideDateTime();
var date2 = new GlideDateTime();
date1.setDisplayValueInternal('2014-01-01 12:00:00');
date2.setDisplayValueInternal('2014-01-01 13:00:00');

// Determine the difference as number of seconds (returns a string)
// Use getDisplayValue() to convert the string to the format expected
by dateDiff()
var diffSeconds = gs.dateDiff(date1.getDisplayValue(),
date2.getDisplayValue(), true);

// JavaScript will coerce diffSeconds from a string to a number
// since diffSeconds is being compared to a number
var msg = (diffSeconds <= 0) ? ' is on or after ' : ' is before ';
gs.print(date1.getDisplayValue() + msg + date2.getDisplayValue())
```

dateGenerate(String, String)

Generates a date and time for the specified date in GMT.

Input Fields

Parameters:

- **Date** - format: yyyy-mm-dd
- **Range** - **start**, **end**, or a time in the 24 hour format **hh:mm:ss**.

Output Fields

Returns: a date and time in the format yyyy-mm-dd hh:mm:ss. If range is **start**, the returned value is yyyy-mm-dd 00:00:00; If range is **end** the return value is yyyy-mm-dd 23:59:59.

daysAgo(int)

Gets a date and time for a certain number of days ago.

Input Fields

Parameters: An integer number of days ago.

Output Fields

Returns: The (GMT) beginning of the days that was the specified number of days ago, in the format **yyyy-mm-dd hh:mm:ss.**

Example

```
function contractNoticeDue() {

    var gr = new GlideRecord("contract");

    gr.addQuery("u_contract_status", "Active");

    gr.query();

    while (gr.next()) {

        if ((gr.u_termination_date <= gs.daysAgo(-90)) && (gr.u_contract_duration == "Long")) {

            gr.u_contract_status = "In review";

        }

        else if ((gr.u_termination_date <= gs.daysAgo(-50)) && (gr.u_contract_duration == "Medium")) {

            gr.u_contract_status = "In review";

        }

        else if ((gr.u_termination_date <= gs.daysAgo(-10)) && (gr.u_contract_duration == "Short")) {

            gr.u_contract_status = "In review";

        }

    }

    gr.update();

}
```

daysAgoEnd(int)

Gets a date and time for end of the day a certain number of days ago.

Input Fields

Parameters: An integer number of days ago.

Output Fields

Returns: The (GMT) end of the day that was the specified number of days ago, in the format **yyyy-mm-dd hh:mm:ss.**

daysAgoStart(int)

Gets a date and time for beginning of the day a certain number of days ago.

Input Fields

Parameters: An integer number of days ago.

Output Fields

Returns: The (GMT) start of the day that was the specified number of days ago, in the format **yyyy-mm-dd hh:mm:ss.**

Example

```
var gr = new GlideRecord('sysapproval_approver');
gr.addQuery('state', 'requested');
gr.addQuery('sys_updated_on', '<', gs.daysAgoStart(5));
gr.query();
```

endOfLastMonth()

Gets the date and time for the end of last month in GMT.

Output Fields

Returns: the GMT end of last month, in the format **yyyy-mm-dd hh:mm:ss.**

endOfLastWeek()

Gets the date and time for the end of last week in GMT, in the format **yyyy-mm-dd hh:mm:ss.**

Output Fields

Returns: the GMT end of last week, in the format **yyyy-mm-dd hh:mm:ss.**

endOfLastYear()

Gets the date and time for the end of last year in GMT.

Output Fields

Returns: the GMT end of last year, in the format **yyyy-mm-dd hh:mm:ss.**

endOfNextMonth()

Gets the date and time for the end of next month in GMT.

Output Fields

Returns: the GMT end of next month, in the format **yyyy-mm-dd hh:mm:ss.**

endOfNextWeek()

Gets the date and time for the end of next week in GMT.

Output Fields

Returns: the GMT end of next week, in the format **yyyy-mm-dd hh:mm:ss.**

endOfNextYear()

Gets the date and time for the end of next year in GMT.

Output Fields

Returns: the GMT end of next year, in the format **yyyy-mm-dd hh:mm:ss.**

endOfThisMonth()

Gets the date and time for the end of this month in GMT.

Output Fields

Returns: the GMT end of this month, in the format (yyyy-mm-dd huh:mm:ss)

endOfThisQuarter()

Gets the date and time for the end of this quarter in GMT.

Output Fields

Returns: the GMT end of this quarter, in the format **yyyy-mm-dd hh:mm:ss.**

endOfThisWeek()

Gets the date and time for the beginning of this week in GMT.

Output Fields

Returns: the GMT beginning of this week, in the format **yyyy-mm-dd hh:mm:ss.**

endOfThisYear()

Gets the date and time for the end of this year in GMT.

Output Fields

Returns: the GMT end of this year, in the format **yyyy-mm-dd hh:mm:ss.**

endOfToday()

Gets the date and time for the end of today in GMT.

Output Fields

Returns: the GMT end of today, in the format **yyyy-mm-dd hh:mm:ss.**

endOfYesterday()

Gets the date and time for the end of yesterday in GMT.

Output Fields

Returns: the GMT end of yesterday, in the format (yyyy-mm-dd huh:mm:ss).

hoursAgo(int)

Gets a date and time for a certain number of hours ago.

Input Fields

Parameters: An integer number of hours ago.

Output Fields

Returns: The (GMT) time that was the specified number of hours ago,in the format **yyyy-mm-dd hh:mm:ss.**

Example

```
if (current.operation() == 'insert') {
 // If no due date was specified, calculate a default
 if (current.due_date == '') {

  if (current.urgency == '1') {
   // Set due date to 4 hours ahead of current time
   current.due_date = gs.hoursAgo(-4);
  }

  if (current.urgency == '2') {
   // Set due date to 2 days ahead of current time
   current.due_date = gs.daysAgo(-2);
  }

  if (current.urgency == '3') {
   // Set due date to 7 days ahead of current time
   current.due_date = gs.daysAgo(-7);
  }
 }
}
```

hoursAgoEnd(int)

Gets a date and time for the end of the hour a certain number of hours ago.

Input Fields

Parameters: An integer number of hours ago.

Output Fields

Returns: The (GMT) end of the hour that was the specified number of hours ago, in the format **yyyy-mm-dd hh:mm:ss.**

hoursAgoStart(int)

Gets a date and time for the start of the hour a certain number of hours ago.

Input Fields

Parameters: An integer number of hours ago.

Output Fields

Returns: The (GMT) start of the hour that was the specified number of hours ago, in the format **yyyy-mm-dd hh:mm:ss.**

lastWeek()

Date and time one week ago in GMT.

Output Fields

Returns: the date and time one week ago, in the format **yyyy-mm-dd hh:mm:ss.**

minutesAgo(int)

Gets a date and time for a certain number of minutes ago.

Input Fields

Parameters: An integer number of minutes ago.

Output Fields

Returns: The (GMT) time that was the specified number of minutes ago, in the format **yyyy-mm-dd hh:mm:ss.**

Example

```
//
// Check to see if the user has failed to login too many times
// when the limit is reached, lock the user out of the system
//
  //Check failed logins in the last 15 minutes
  var gr = new GlideRecord('sysevent');
  gr.addQuery('name', 'login.failed');
  gr.addQuery('parm1', event.parm1.toString());
  gr.addQuery('sys_created_on','>=', gs.minutesAgo(15));
  gr.query();
  var rowCount = gr.getRowCount();
  if(rowCount >= 5){
      var gr = new GlideRecord("sys_user");
      gr.addQuery("user_name", event.parm1.toString());
      gr.query();
      if (gr.next()) {
           gr.locked_out = true;
           gr.update();
           gs.log("User " + event.parm1 + " locked out due to too many
invalid login attempts");
      }
  }
```

minutesAgoEnd(int)

Gets a date and time for the end of the minute a certain number of minutes ago.

Input Fields

Parameters: An integer number of minutes ago.

Output Fields

Returns: The (GMT) end of the minute that was the specified number of minutes ago, in the format **yyyy-mm-dd hh:mm:ss.**

minutesAgoStart(int)

Gets a date and time for the start of the minute a certain number of minutes ago.

Input Fields

Parameters: An integer number of minutes ago.

Output Fields

Returns: The (GMT) start of the minute that was the specified number of minutes ago, in the format **yyyy-mm-dd hh:mm:ss.**

monthsAgo(int)

Gets a date and time for a certain number of months ago.

Input Fields

Parameters: An integer number of months ago.

Output Fields

Returns: A value (GMT) on the current (today's) date of the specified month, in the format **yyyy-mm-dd hh:mm:ss.**

monthsAgoEnd(int)

Gets a date and time for the last day of the month a certain number of months ago.

Input Fields

Parameters: An integer number of months ago.

Output Fields

Returns: The (GMT) end of the month that was the specified number of months ago, in the format **yyyy-mm-dd hh:mm:ss.**

monthsAgoStart(int)

Gets a date and time for the start of the minute a certain number of minutes ago.

Input Fields

Parameters: An integer number of minutes ago.

Output Fields

Returns: The (GMT) start of the minute that was the specified number of minutes ago, in the format **yyyy-mm-dd hh:mm:ss.**

now()

Gets the current date using GMT.

Output Fields

Returns: The current date in the user defined format, according to GMT.

Example

```
// When the user password changes then set the u_password_last_reset
field
// to now so we know when to force another update

var gr = new GlideRecord("sys_user");
if (gr.get(event.parm1.toString())) {
    // Do something based on the Password Changing
    gs.log("The user password changed so do something else...");
    gr.u_password_last_reset = gs.now();
    gr.update();
}
```

nowNoTZ()

Gets the current date and time in UTC format.

Output Fields

Returns: the current UTC date time.

Example

```
// When the user password changes then set the u_password_last_reset
field
// to now so we know when to force another update

var gr = new GlideRecord("sys_user");
if (gr.get(event.parm1.toString())) {
    // Do something based on the Password Changing
    gs.log("The user password changed so do something else...");
    gr.u_password_last_reset = gs.nowNoTZ();
    gr.update();
}
```

nowDateTime()

Gets the current date and time in the user-defined format.

Output Fields

Returns: The current date and time in the user-defined format.

Example

The following script sets the field **u_target_date** to the current date and time:

```
current.u_target_date = gs.nowDateTime();
```

After the script is run, the **u_target_date** field will hold the date and time of the moment the script is run, in the system format.

Example 2

When setting a variable in a workflow script to the current date and time, use the setDisplayValue method. The following script sets the workflow variable **end_date** to the current date and time:

```
current.variables.end_date.setDisplayValue(gs.nowDateTime());
```

After the script is run, the **end_date** field will hold the date and time of the moment the script is run, in the system format.

quartersAgo(int)

Gets a date and time for a certain number of quarters ago.

Input Fields

Parameters: An integer number of quarters ago.

Output Fields

Returns: The (GMT) beginning of the quarter that was the specified number of quarters ago, in the format **yyyy-mm-dd hh:mm:ss.**

quartersAgoEnd(int)

Gets a date and time for the last day of the quarter a certain number of quarters ago.

Input Fields

Parameters: An integer number of quarters ago.

Output Fields

Returns: The (GMT) end of the quarter that was the specified number of quarters ago, in the format **yyyy-mm-dd hh:mm:ss.**

quartersAgoStart(int)

Gets a date and time for the first day of the quarter a certain number of quarters ago.

Input Fields

Parameters: An integer number of quarters ago.

Output Fields

Returns: The (GMT) end of the month that was the specified number of quarters ago, in the format **yyyy-mm-dd hh:mm:ss.**

yearsAgo(int)

Gets a date and time for a certain number of years ago.

Input Fields

Parameters: An integer number of years ago.

Output Fields

Returns: The (GMT) beginning of the years that was the specified number of years ago, in the format **yyyy-mm-dd hh:mm:ss.**

yesterday()

Gets yesterday's time.

Output Fields

Returns: The (GMT) for 24 hours ago, in the format **yyyy-mm-dd hh:mm:ss.**

isFirstDayOfMonth(Object)

Checks whether the date is the first day of the month.

Input Fields

Parameters: date object.

Output Fields

Returns: true if date is the first day of the month, false if it is not.

isFirstDayOfWeek(Object)

Checks whether the date is the first day of the week. This uses the ISO standard of Monday being the first day of the week.

Input Fields

Parameters: date object.

Output Fields

Returns: true if date is the first day of the week, false if it is not.

isFirstDayOfYear(Object)

Checks whether the date is the first day of the year.

Input Fields

Parameters: date object.

Output Fields

Returns: true if date is the first day of the year, false if it is not.

isLastDayOfMonth(Object)

Checks whether the date is the last day of the month.

Input Fields

Parameters: date object.

Output Fields

Returns: true if date is the last day of the month, false if it is not. This uses the ISO standard of Sunday being the last day of the week.

isLastDayOfWeek(Object)

Checks whether the date is the last day of the week.

Input Fields

Parameters: date object.

Output Fields

Returns: true if date is the last day of the week, false if it is not.

isLastDayOfYear(Object)

Checks whether the date is the last day of the year.

Input Fields

Parameters: date object.

Output Fields

Returns: true if date is the last day of the year, false if it is not.

User Session

addErrorMessage(Object)

Adds an error message for the current session. Use getErrorMessages() to retrieve a list of error messages currently being shown.

Input Fields

Parameters: an error object.

Example

```
 gs.include("PrototypeServer");
  var ValidatePasswordStronger = Class.create();
  ValidatePasswordStronger.prototype = {
       process : function() {
          var user_password = request.getParameter("user_password");
          var min_len = 8;
          var rules = "Password must be at least " + min_len +
             " characters long and contain a digit, an uppercase
letter, and a lowercase letter.";
          if (user_password.length() < min_len) {
             gs.addErrorMessage("TOO SHORT: " + rules);
             return false;
          }
          var digit_pattern = new RegExp("[0-9]", "g");
          if (!digit_pattern.test(user_password)) {
             gs.addErrorMessage("DIGIT MISSING: " + rules);
```

```
            return false;
         }
         var upper_pattern = new RegExp("[A-Z]", "g");
         if (!upper_pattern.test(user_password)) {
            gs.addErrorMessage("UPPERCASE MISSING: " + rules);
            return false;
         }
         var lower_pattern = new RegExp("[a-z]", "g");
         if (!lower_pattern.test(user_password)) {
            gs.addErrorMessage("LOWERCASE MISSING: " + rules);
            return false;
         }
         return true; // password is OK
      }
   }
```

addInfoMessage(Object)

Adds an info message for the current session. Use getInfoMessages() to retrieve the list of info messages being shown. **Note**: This method is not supported for asynchronous business rules and cannot be used within transform scripts.

Input Fields

Parameters: an info message object.

Example

```
if ((!current.u_date1.nil()) && (!current.u_date2.nil())) {
  var start = current.u_date1.getGlideObject().getNumericValue();
  var end = current.u_date2.getGlideObject().getNumericValue();
  if (start > end) {
    gs.addInfoMessage('start must be before end');
    current.u_date1.setError('start must be before end');
    current.setAbortAction(true);
  }
}
```

addMessage(String, Object)

Adds a message for the current session. Can be called using getMessages(String).

Input Fields

Parameters:

- String type of message
- Message to store

Example

```
gs.include("FormInfoHeader");
var fi = new FormInfoHeader();
var s = 'An incident ' + current.number + ' has been opened for your
request.<br/>';
s += 'The IT department will contact you when the password is reset or
for further information.<br/>';
//s += 'You can track status from the <a href="home.do" class="breadcrumb" >Home Page</a> <br/>';
fi.addMessage(s);
producer.redirect = 'home.do?sysparm_view=ess';
```

flushMessages()

Clears session messages saved using addErrorMessage(Object) or addInfoMessage(Object). Session messages are shown at the top of the form. In client side scripts use g_form.clearMessages() to remove session messages.

Output Fields

Returns: void

Example

```
gs.flushMessages();
```

getErrorMessages()

Gets the list of error messages for the session that were added by addErrorMessage(Object)

Output Fields

Returns: list of error messages.

getImpersonatingUserDisplayName()

Returns the display name of the impersonating user.

Output Fields

Returns: display name of impersonating user.

getImpersonatingUserName()

Returns the name of the impersonating user or null if not impersonating

Output Fields

Returns: name of the impersonating user.

getInfoMessages()

Gets the list of info messages for the session that were added via addInfoMessage(Object).

Output Fields

Returns: the list of info messages.

getMessages(String)

Gets the list of messages of the specified type for the session that were added via addMessage(String, Object).

Input Fields

Parameters: string type of message.

Output Fields

Returns: list of messages of string type.

getNodeValue(object, Integer)

Gets the node value for specified index.

Input Fields

Parameters

- Object to examine.
- Integer for the index to get a node value from.

Output Fields

Returns: the node's value.

getNodeName(Object, Integer)

Returns the node name for specified index.

Input Fields

Parameters

- Object to examine.
- Integer for the index to get a node value from.

Output Fields

Returns: the node's name.

getPreference(String, Object)

Gets a user preference.

Input Fields

Parameters:

- String key for the preference.
- Object default value

Output Fields

Returns: a string value for the preference. If null, uses the default value specified above.

getSession()

Returns a GlideSession object. See the GlideSession APIs wiki page for methods to use with the GlideSession object.

Output Fields

Returns: a GlideSession object for the current session.

Example

```
var session = gs.getSession();
```

getSessionID()

Accesses the GlideSession Session ID.

Output Fields

Returns: the Session ID.

getTrivialMessages()

Gets the list of error messages for the session that were added with the trivial flag.

Output Fields

Returns: the list of messages.

getUser()

Returns a reference to the User object for the current user. More information is available here.

Output Fields

Returns: a reference to a User object.

Example

```
var myUserObject = gs.getUser()
```

getUserDisplayName()

Returns the name field of the current user (e.g. John Smith, as opposed to smith).

Output Fields

Returns: the name field of the current user (e.g. John Smith, as opposed to jsmith).

Example

```
    <g2:evaluate var="jvar_current_user" expression="gs.getUserDisplayName()"/>
     $[JS:jvar_current_user]
```

getUserID()

Returns the sys_id of the current user.

Output Fields

Returns: the sys_id of the current user.

Example

```
if (current.operation() != 'insert'&& current.comments.changes()) {
    gs.eventQueue("incident.commented", current, gs.getUserID(),
gs.getUserName());
}
```

getUserName()

Returns the username of the current user (e.g., jsmith).

Output Fields

Returns: the username of the current user (e.g., jsmith).

Example

```
//Add a comment when closing
    current.comments = "Closed by " + gs.getUserName() + " at " +
gs.nowDateTime();
    gs.addInfoMessage("Close comment added");
}
```

getUserNameByUserID(String)

Gets the username based on a user ID.

Input Fields

Parameters: a sys_id as a string.

Output Fields

Returns: the username.

getXMLText (String, String)

Gets the xml text for the first node in the xml string that matches the xpath query.

Input Fields

Parameters:

- XML to search within
- xpath query to match

Output Fields

Returns: the XML node value as text.

getXMLNodeList(String)

Constructs an Array of all the nodes and values in an XML document.

Input Fields

Parameters: XML document to parse.

Output Fields

Returns: an array list of names and values.

hasRole(String)

Determines if the current user has the specified role. This method returns true for users with the administrator role.

Input Fields

Parameters: the role to check.

Output Fields

Returns: true if the user has the role or false if the user does not have the role.

Example

```
if (!gs.hasRole("admin") && !gs.hasRole("groups_admin") &&
gs.getSession().isInteractive()) {
  var qc = current.addQuery("u_hidden", "!=", "true"); //cannot see
hidden groups...
```

```
  qc.addOrCondition("sys_id", "javascript:getMyGroups()"); //...unless
in the hidden group
}
```

hasRoleInGroup(Object, Object)

Determines if the current user has the specified role within a specified group.

Input Fields

Parameters:

- The name of the role to check for.
- A GlideRecord or the sys_id of the group to check within.

Output Fields

Returns: Returns true if all of the following conditions are met:

1. The logged-in user HAS the role in question
2. The "Granted by" field on the user role record is set to the specified group
3. The "inherited" field on the user role record is false

Example

```
var group = new GlideRecord('sys_user_group');
group.addQuery('name', 'GROUP_NAME');
group.setLimit(1);
group.query();
if (group.next()) {
   if (gs.hasRoleInGroup('role_name', group)) {
      gs.print('User has role in group');
   } else {
      gs.print('User does NOT have role in group');
   }
}
```

isInteractive()

Checks if the current session is interactive. An example of an interactive session is when a user logs in using the log-in screen. An example of a non-interactive session is using a SOAP request to retrieve data.

Output Fields

Returns: true if the session is interactive.

Example

```
if (!gs.hasRole('admin') && gs.isInteractive()) {
    var qc1 = current.addQuery('u_group', '');
    var gra = new GlideRecord('sys_user_grmember');
    gra.addQuery('user', gs.getUserID());
    gra.query();
    while (gra.next()) {
      qc1.addOrCondition('u_group', gra.group);
   }
}
```

isLoggedIn()

Determines if the current user is currently logged in.

Output Fields

Returns: true if the current user is logged in, false if the current user is not.

setRedirect(Object)

Sets the redirect URI for this transaction. This determines the next page the user will see.

Input Fields

Parameters: URI object to set as redirect.

Example

The following example redirects the user to a particular Catalog Item, and passes along the current email as a parameter:

```
gs.setRedirect("com.glideapp.servicecatalog_cat_item_view.do?sysparm_id=d41ce5bac611227a0167f4bf8109bf70&sysparm_user="


+ current.sys_id + "&sysparm_email=" + current.email)
```

setReturn(Object)

Sets the return URI for this transaction. This determines what page the user will be directed to when they return from the next form.

Input Fields

Parameters: URI object to set as return.

Example

The following example ensures that the user will be returned to the current page when they are done with the next one.

```
   gs.setReturn (current.getLink(true));
```

userID()

Returns the sys_id of the user associated with this session. A shortcut for the more proper getUserID().

Output Fields

Returns: sys_id of current user.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/glide-server-apis/concept/c_GlideSystem.html
[2] http://docs.oracle.com/javase/6/docs/api/?java/text/MessageFormat.html

GlideElement

Note: *This article applies to Fuji. For more current information, see Server API Reference* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

GlideElement provides a number of convenient script methods for dealing with fields and their values. GlideElement methods are available for the fields of the current GlideRecord.

Note: *The global Glide APIs are not available in scoped scripts. There are scoped Glide APIs for use in scoped scripts. The scoped Glide APIs provide a subset of the global Glide API methods. For information on scoped applications see Application Scope, scripting in scoped applications, and the list of scoped APIs.*

Method Summary

Method Summary	Description
canCreate()	Determines if the user's role permits creation of new records in this field.
canRead()	Determines if the GlideRecord table can be read from.
canWrite()	Determines if the GlideRecord table can be written to.
changes()	Determines if the current field has been modified.
changesFrom(Object)	Determines the previous value of the current field matched a certain object.
changesTo(Object)	Determines if the new value of a field after a change matches a certain object.
debug(Object)	Debugs the object and adds debug messages using setError(String).
getAttribute(String)	Gets the value of the attribute on the field in question from the dictionary as a string. If the attribute is a boolean attribute, use getBooleanAttribute(String) to get the value as a boolean rather than as a string.
getBaseTableName()	Gets the base table of the field.
getBooleanAttribute(String)	Gets the value of the attribute on the field in question from the dictionary as a boolean. To get the value as a string, use getAttribute(string).
getChoices(String)	Generates a choice list for a field. Returns the choice values from the base table only, not from the extended table.
getChoiceValue()	Gets the choice label for the current choice value.

getDebugCount()	Gets the number of debug messages logged by debug()
getDependent()	Gets the field that this field is dependent on.
getDependentTable()	Gets the table that the current table depends on.
getDisplayValue(Int)	Gets the formatted display value of the field.
getDisplayValueExt(Int, String)	Gets the formatted display value of a field, or a specified substitute value if the display value is null or empty.
getED()	Gets an element descriptor.
getElementValue(String)	Gets the value for a given element.
getError()	Gets error debug messages.
getEscapedValue()	Gets the escaped value for the current element.
getFieldStyle()	Gets the CSS style for the field.
getGlideObject()	Gets a glide object.
getGlideRecord()	Gets the GlideRecord object that contains the element. To get a GlideRecord for a reference element, use getRefRecord().
getHTMLValue(Int)	Gets the HTML value of a field.
getHTMLValueExt(Int, String)	Gets the HTML value of a field, or a specified substitute value if the HTML value is null or empty.
getJournalEntry(int)	Gets either the most recent journal entry or all journal entries.
getLabel()	Gets the object's label.
getName()	Gets the name of the field.
getRefRecord()	Gets a GlideRecord object for a given reference element.
getStyle()	Get a CSS style for the value.
getTableName()	Gets the name of the table the field is on.
getTextAreaDisplayValue()	Gets the value and escapes the HTML.
getXHTMLValue()	Gets the XHTML value of a field as a string.
getXMLValue()	Gets the XML value of a field as a string.
hasAttribute(String)	Determines whether a field has a particular attribute.
hasRightsTo(String)	Determines if the user has the right to perform a particular operation.
hasValue()	Determines if the field has a value.
nil()	Determines whether the field is null.
setDisplayValue(Object)	Sets the display value of the field.
setError(String)	Adds an error message. Can be retrieved using getError().
setInitialValue(String)	Sets the initial value of a field.
setJournalEntry(Object, String)	Sets a journal entry.
setValue(Object)	Sets the value of a field.
toString()	Converts the value to a string.

Method Detail

canCreate()

Determines if the user's role permits creation of new records in this field.

Output Fields

Returns: true if the field can be created, false if the field cannot.

canRead()

Determines if the GlideRecord table can be read from.

Output Fields

Returns: true if the field can be read, false if the field cannot.

canWrite()

Determines if the GlideRecord table can be written to.

Output Fields

Returns: true if the field can be written to, false if the field cannot.

changes()

Determines if the current field has been modified, if the field is a reference, integer, or string field. Note that changes to Journal fields are not detected by this method.

Output Fields

Returns: true if the fields have been changed, false if the field has not.

changesFrom(Object)

Determines the previous value of the current field matched a certain object.

Input Fields

Parameters: an object value to check against the previous value of the current field.

Output Fields

Returns: true if the field's previous value matches, false if it does not.

Example

```
if (theState.changesTo(resolvedState)) {
  operation = 4; //Resolved
}
else if (theState.changesTo(closedState)) {
  operation = 11; //Resolution Accepted
}
else if (theState.changesFrom(resolvedState) ||
theState.changesFrom(closedState)) {
  operation = 10; //Re-open
}
else {
  operation = 6; //Update
}
```

changesTo(Object)

Determines if the new value of a field after a change matches a certain object.

Input Fields

Parameters: an object value to check against the new value of the current field.

Output Fields

Returns: true if the field's new value matches, false if it does not.

Example

```
if (theState.changesTo(resolvedState)) {
  operation = 4; //Resolved
}
else if (theState.changesTo(closedState)) {
  operation = 11; //Resolution Accepted
}
else if (theState.changesFrom(resolvedState) ||
theState.changesFrom(closedState)) {
  operation = 10; //Re-open
}
else {
  operation = 6; //Update
}
```

debug(Object)

Debugs the object and adds debug messages using setError(String).

Input Fields

Parameters: an object to debug.

getAttribute(String)

Gets the value of the attribute on the field in question from the dictionary as a string. If the attribute is a boolean attribute, use getBooleanAttribute(String) to get the value as a boolean rather than as a string.

Input Fields

Parameters: a string attribute name to get a value from.

Output Fields

Returns: the attribute's value as a string.

Example

```
doit();
function doit() {
  var gr = new GlideRecord('sys_user');
  gr.query("user_name","admin");
  if (gr.next()) {
    gs.print("we got one");
    gs.print(gr.location.getAttribute("tree_picker"));
  }

}
```

getBaseTableName()

Gets the base table of the field.

Note: *This may be different from the table that the field is defined on. See Tables and Classes.*

Output Fields

Returns: name of the base table.

getBooleanAttribute(String)

Gets the value of the attribute on the field in question from the dictionary as a boolean. To get the value as a string, use getAttribute(string).

Input Fields

Parameters: the string name of the attribute to get a value from.

Output Fields

Returns: the boolean value of the string.

getChoices(String)

Generates a choice list for a field. Returns the choice values from the base table only, not from the extended table.

Input Fields

Parameters: a dependent value (optional).

Output Fields

Returns: an array list of choices.

Example

```
var field = gr.getElement('os');
var choices = field.getChoices();
```

getChoiceValue()

Gets the choice label for the current choice value.

Output Fields

Returns: a string choice label.

getDebugCount()

Gets the number of debug messages logged by debug()

Output Fields

Returns: an integer number of debug messages.

getDependent()

Checks whether or not the field is dependent on another field.

Output Fields

Returns: the name of the field the current field depends on.

getDependentTable()

Gets the table that the current table depends on.

Output Fields

Returns: the string name of the table.

getDisplayValue(Int)

Gets the formatted display value of the field.

Input Fields

Parameters: an integer number of maximum characters desired (optional).

Output Fields

Returns: the display value of the field.

Example

```
var fields = current.getFields();
for (var i = 0; i < fields.size(); i++) {
  var field = fields.get(i);
  var name = field.getName();
  var value = field.getDisplayValue();
  gs.print(i + ". " + name + "=" + value);
}
```

getDisplayValueExt(Int, String)

Gets the formatted display value of a field, or a specified substitute value if the display value is null or empty.

Input Fields

Parameters:

- **maxCharacters** - (int) the number of maximum characters desired (optional).
- **nullSub** - (String) the value to return if the display value is null or empty.

Output Fields

Returns: (String) the formatted display value of a field, or a specified substitute value

getED()

Gets an element descriptor.

Output Fields

Returns: the element descriptor.

Example

```
var fields = current.getFields();
for (i=0; i<fields.size(); i++) {
  var field = fields.get(i);
  var descriptor = field.getED();
  gs.print("type=" + descriptor.getType() +
    " internalType=" + descriptor.getInternalType());
}
```

getElementValue(String)

Gets the value for a given element.

Input Fields

Parameters: an element to get a value from.

Output Fields

Returns: the value of the element.

getError()

Gets error debug messages.

Output Fields

Returns: a string of debug messages.

getEscapedValue()

Gets the escaped value for the current element.

Output Fields

Returns: the escaped value of the current element.

getFieldStyle()

Gets the CSS style for the field.

Output Fields

Returns: the CSS style for the field as a string.

Example

```
var fields = current.getFields();
for (var i = 0; i < fields.size(); i++) {
  var field = fields.get(i);
  var name = field.getName();
  var value = field.getDisplayValue();
  gs.print(i + ". " + name + "=" + value);
}
```

getGlideObject()

Gets a glide object.

Output Fields

Returns: a glide object.

Example

Note: *This API call changed in the Calgary release:*

- *GlideCalendar* replaces *Packages.com.glide.schedule.GlideCalendar*

The new script object calls apply to the Calgary release and beyond. For releases prior to Calgary, substitute the packages call as described above. Packages calls are not valid beginning with the Calgary release. For more information, see Scripting API Changes.

```
function calcDateDelta(start, end, calendar) {
  var cal = GlideCalendar.getCalendar(calendar);
  if (!cal.isValid())
      return null;
  var realStart = start.getGlideObject();
  var realEnd = end.getGlideObject();
  var duration = cal.subtract(realStart, realEnd);
  return duration;
}
```

getGlideRecord()

Gets a glide record.

Output Fields

Returns: a glide record.

Example

```
function task_ci_allocate() {
   var cnt = g_list.getRowCount();
   if (cnt == 0)
      return;

   var pct = 100.0 / cnt;
   var pct = (parseInt((pct + .005) * 100)) / 100;
   var gr = g_list.getGlideRecord();
   gr.query();
   while (gr.next()) {
      gr.u_allocation = pct;
      gr.update();
   }
}
```

getHTMLValue(Int)

Gets the HTML value of a field.

Parameters:

- **maxChars** - (int) the number of maximum characters desired (optional).

Output Fields

Returns: an integer value of maximum characters to limit the output.

Example

```
var inccause = new GlideRecord("incident");
inccause.short_description = current.short_description;
inccause.comments = current.comments.getHTMLValue();
inccause.insert();
```

getHTMLValueExt(Int, String)

Gets the HTML value of a field, or a specified substitute value if the HTML value is null or empty.

Input Fields

Parameters:

- **maxCharacters** - (int) the number of maximum characters desired (optional).
- **nullSub** - (String) the value to return in the HTML value is null or empty.

Output Fields

Returns: the HTML value of a field, or a specified substitute value.

getJournalEntry(int)

Gets either the most recent journal entry or all journal entries.

Parameters: -1: get all journal entries, 1: get the most recent journal entry.

Output Fields

Returns: (Sting) For the most recent entry, returns a sting that contains the field label, timestamp, and user display name of the journal entry. For all journal entries, returns the same information for all journal entries ever entered as a single string with each entry delimited by "\n\n".

Example

```
var notes = current.work_notes.getJournalEntry(-1); //gets all journal
entries as a string where each entry is delimited by '\n\n'
var na = notes.split("\n\n");                        //stores each entry
 into an array of strings

for (var i = 0; i < na.length; i++)
  gs.print(na[i]);
```

getLabel()

Gets the object's label.

Output Fields

Returns: the label as a string.

Example

```
var gr = new GlideRecord("sc_req_item");
gr.addQuery("request", current.sysapproval);
gr.query();
while(gr.next()) {
    var nicePrice = gr.price.toString();
    if (nicePrice != ) {
        nicePrice = parseFloat(nicePrice);
        nicePrice = nicePrice.toFixed(2);
    }
    template.print(gr.number + ":  " + gr.quantity + " X " +
gr.cat_item.getDisplayValue() + " at $" + nicePrice + " each \n");
    template.print("    Options:\n");
    for (key in gr.variables) {
      var v = gr.variables[key];
      if(v.getGlideObject().getQuestion().getLabel() != ) {
         template.space(4);
         template.print('     ' +
v.getGlideObject().getQuestion().getLabel() + " = " +
v.getDisplayValue() + "\n");
      }
    }
}
```

getName()

Gets the name of the field.

Output Fields

Returns: string name of the field.

Example

Note: *This API call changed in the Calgary release:*

- *GlideMutex* replaces *Packages.com.glide.sys.lock.Mutex*

The new script object calls apply to the Calgary release and beyond. For releases prior to Calgary, substitute the packages call as described above. Packages calls are not valid beginning with the Calgary release. For more information, see Scripting API Changes.

```
var lock_name = "my_lock";
gs.print("getting lock");
var lock = new GlideMutex.get(lock_name);
    //.get(lock_name) gets an exclusive lock; replace with
.getNonExclusive(lock_name) for a non-exclusive lock)
while (lock.toString()+'' == "undefined") {''
   gs.print("waiting for lock: " + lock_name);
   gs.sleep(1000); // sleep for 1000 milliseconds
   lock = new GlideMutex.get(lock_name);
}
gs.print("got lock for " + lock.getName());
//***********************************
// do your activity requiring a lock here
//***********************************
lock.release();
gs.print("released lock");
```

getRefRecord()

Gets a GlideRecord object for a given reference element.

Output Fields

Returns: a GlideRecord object.

Example

```
var grINC = new GlideRecord('incident');
grINC.notNullQuery('caller_id');
grINC.query();
if (grINC.next()) {

  // Get a GlideRecord object for the referenced sys_user record
  var grUSER = grINC.caller_id.getRefRecord();
  gs.print( grUSER.getValue('name') );
}
```

getStyle()

Get a CSS style for the value.

Output Fields

Returns: the CSS style for the value.

Example

```
// Get string of style field from Field Style record, if applicable
var cssStyle = gr.state.getStyle();
```

getTableName()

Gets the name of the table the field is on.

Note: *This may be different from the table class that the record is in. See Tables and Classes.*

Output Fields

Returns: string name of the table.

Example

```
 if (current.getTableName() == "sysapproval_approver") {
  if (current.approver == email.from_sys_id)  {
     current.comments = "reply from: " + email.from + "\n\n" +
email.body_text;

   // if it's been cancelled, it's cancelled.
  var doit = true;
  if (current.state=='cancelled')
      doit = false;

  if (email.body.state != undefined)
     current.state= email.body.state;

   if (doit)
      current.update();
} else {
   gs.log("Approval for task
("+current.sysapproval.getDisplayValue()+") rejected because user
sending
           email( "+email.from+") does not match the approver
("+current.approver.getDisplayValue()+")");
}
}
```

getTextAreaDisplayValue()

Gets the value and escapes the HTML.

Output Fields

Returns: the string value with the HTML escaped.

getXHTMLValue()

Gets the XHTML value of a field as a string.

Output Fields

Returns: the XHTML value as a string.

getXMLValue()

Gets the XML value of a field as a string.

Output Fields

Returns: the XML value as a string.

hasAttribute(String)

Determines whether a field has a particular attribute.

Input Fields

Parameters: the string attribute to check for.

Output Fields

Returns: true if the field has the attribute, false if not.

Example

```
var totalCritical = 0;

var filledCritical = 0; var fields = current.getFields();
gs.print(fields); for (var num = 0; num &lt; fields.size(); num++) {

    gs.print("RUNNING ARRAY VALUE " + num);
   var ed = fields.get(num).getED();
   if(ed.hasAttribute("tiaa_critical")) {
       gs.print("CRITICAL FIELD FOUND");
       totalCritical ++;
       if (!fields.get(num).isNil()) {
           filledCritical ++;
       }
   }

} var answer = 0; gs.print("TOTAL - " + totalCritical);
gs.print("FILLED - " + filledCritical); if (filledCritical &gt; 0
&amp;&amp; totalCritical &gt; 0){

    var pcnt = (filledCritical/totalCritical)*100;
   answer = pcnt.toFixed(2);;

} answer;
```

hasRightsTo(String)

Determines if the user has the right to perform a particular operation.

Input Fields

Parameters: the string name of the operation to check for.

Output Fields

Returns: true of the user has the right, false if the user does not.

hasValue()

Determines if the field has a value.

Output Fields

Returns: true if the field has a value, false if not.

nil()

Determines whether the field is null.

Output Fields

Returns: true if the field is null or an empty string, false if not.

Example

```
 if (current.start_date.changes() || current.end_date.changes() ||
current.assigned_to.changes()) {
  if (!current.start_date.nil() && !current.end_date.nil() &&
!current.assigned_to.nil()) {
 gs.eventQueue("change.calendar.notify", current, current.assigned_to,
previous.assigned_to);

}
```

setDisplayValue(Object)

Sets the display value of the field.

Input Fields

Parameters: the object to serve as the display value.

Example

```
current.assignment_group.setDisplayValue('Network');
```

setError(String)

Adds an error message. Can be retrieved using getError().

Input Fields

Parameters: a string error message.

Example

```
if ((!current.u_date1.nil()) && (!current.u_date2.nil())) {
  var start = current.u_date1.getGlideObject().getNumericValue();
  var end = current.u_date2.getGlideObject().getNumericValue();
  if (start > end) {
    gs.addInfoMessage('start must be before end');
    current.u_date1.setError('start must be before end');
    current.setAbortAction(true);
  }
}
```

setInitialValue(String)

Sets the initial value of a field.

Input Fields

Parameters: a string value of a field.

setJournalEntry(Object, String)

Sets a journal entry. The username parameter does not set the journal entry's created by field.

Input Fields

Parameters:

- The value to set to the journal entry.
- The username to attribute the journal entry to. Does not set the journal entry's created by field.

setValue(Object)

Sets the value of a field.

Input Fields

Parameters: object value to set the field to.

Example

```
// Using GlideElement.setValue (equivalent to GlideRecord.setValue)
gr.short_description.setValue('This is the short description.');

// Using GlideRecord.setValue
gr.setValue('short_description', 'This is the short description.');
```

toString()

Converts the value to a string.

Output Fields

Returns: the value as a string.

Example

```
doit();

function doit() {

  var gr = new GlideRecord('sys_user');
  gr.query();
  while (gr.next()) {
    if ((gr.first_name.toString().length !=
gr.first_name.toString().trim().length) ||
(gr.last_name.toString().length
         != gr.last_name.toString().trim().length)) {
      gr.first_name = gr.first_name.toString().trim();
      gr.last_name = gr.last_name.toString().trim();
      gr.autoSysFields(false);
      gr.update();
    }
  }

}
```

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/server-api/topic/p_ServerAPIReference.html

GlideAggregate

Note: *This article applies to Fuji. For more current information, see GlideAggregate* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Note: This functionality requires a knowledge of **JavaScript**.

Overview

The GlideAggregate class is an extension of GlideRecord and allows database aggregation (COUNT, SUM, MIN, MAX, AVG) queries to be done. This can be helpful in creating customized reports or in calculations for calculated fields. GlideAggregation is an extension of GlideRecord and its use is probably best shown through a series of examples.

Note: *The global Glide APIs are not available in scoped scripts. There are scoped Glide APIs for use in scoped scripts. The scoped Glide APIs provide a subset of the global Glide API methods. For information on scoped applications see Application Scope, scripting in scoped applications, and the list of scoped APIs.*

Examples

Here is an example that simply gets a count of the number of records in a table:

```
var count = new GlideAggregate('incident');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next())
   incidents = count.getAggregate('COUNT');
```

There is no query associated with the above example but if you wanted to get a count of the incidents that were open then you simply add a query just as is done with GlideRecord. Here is an example to get a count of the number of active incidents.

```
var count = new GlideAggregate('incident');
count.addQuery('active', 'true');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next())
   incidents = count.getAggregate('COUNT');
```

To get a count of all of the open incidents by category the code is:

```
var count = new GlideAggregate('incident');
count.addQuery('active', 'true');
count.addAggregate('COUNT', 'category');
count.query();
while (count.next()) {
   var category = count.category;
   var categoryCount = count.getAggregate('COUNT', 'category');
   gs.log("The are currently " + categoryCount + " incidents with a
category of " + category);
}
```

The output is:

```
       *** Script: The are currently 1.0 incidents with a category of Data
       *** Script: The are currently 11.0 incidents with a category of Enhancement
       *** Script: The are currently 1.0 incidents with a category of Implementation
       *** Script: The are currently 197.0 incidents with a category of inquiry
       *** Script: The are currently 13.0 incidents with a category of Issue
       *** Script: The are currently 1.0 incidents with a category of
       *** Script: The are currently 47.0 incidents with a category of request
```

Below is an example that shows that you can ask for multiple aggregations. We are asking to see how many times records have been modified and we want the MIN, MAX, and AVG values.

```
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
   var min = count.getAggregate('MIN', 'sys_mod_count');
   var max = count.getAggregate('MAX', 'sys_mod_count');
   var avg = count.getAggregate('AVG', 'sys_mod_count');
   var category = count.category.getDisplayValue();
   gs.log(category + " Update counts: MIN = " + min + " MAX = " + max +
 " AVG = " + avg);
}
```

The output is:

```
       *** Script: Data Import Update counts: MIN = 4.0 MAX = 21.0 AVG = 9.3333
       *** Script: Enhancement Update counts: MIN = 1.0 MAX = 44.0 AVG = 9.6711
       *** Script: Implementation Update counts: MIN = 4.0 MAX = 8.0 AVG = 6.0
       *** Script: inquiry Update counts: MIN = 0.0 MAX = 60.0 AVG = 5.9715
       *** Script: Inquiry / Help Update counts: MIN = 1.0 MAX = 3.0 AVG = 2.0
       *** Script: Issue Update counts: MIN = 0.0 MAX = 63.0 AVG = 14.9459
       *** Script: Monitor Update counts: MIN = 0.0 MAX = 63.0 AVG = 3.6561
       *** Script: request Update counts: MIN = 0.0 MAX = 53.0 AVG = 5.0987
```

Here is a somewhat more complex example that shows how to compare activity from one month to the next.

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
     last = agg2.getAggregate('count','category');
  }
  gs.log(category + ": Last month:" + count + " Previous Month:" +
last);

}
```

The output is:

```
*** Script: Monitor: Last month:6866.0 Previous Month:4468.0
 *** Script: inquiry: Last month:142.0 Previous Month:177.0
 *** Script: request: Last month:105.0 Previous Month:26.0
 *** Script: Issue: Last month:8.0 Previous Month:7.0
 *** Script: Enhancement: Last month:5.0 Previous Month:5.0
 *** Script: Implementation: Last month:1.0 Previous Month:0
```

Here is an example to get distinct count of a field on a group query.

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count');
agg.addAggregate('count(distinct','category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
//
agg.groupBy('priority');
agg.query();
while (agg.next()) {
// Expected count of incidents and count of categories within each
```

```
priority value (group)
gs.info('Incidents in priority ' + agg.priority + ' = ' +
agg.getAggregate('count') +
            ' (' + agg.getAggregate('count(distinct','category') + '
categories)');
}
```

The output is:

```
*** Script: Incidents in priority 1 = 13 (3 categories)
*** Script: Incidents in priority 2 = 10 (5 categories)
*** Script: Incidents in priority 3 = 5 (3 categories)
*** Script: Incidents in priority 4 = 22 (6 categories)
```

Method Summary

Method Summary

Return Value	Details
void	**addEncodedQuery**(`String query`) Adds a query to the Aggregate. Adds an encoded query to the other queries that may have been set for this aggregate.
void	**addHaving**(`String name, String operator, String value`) Adds a "having" element to the aggregate e.g. select category, count(*) from incident group by category HAVING count(*) > 5
void	**addAggregate**(`String agg, String name`) Adds an aggregate.
void	**addTrend**(`String fieldName, String timeInterval`) Adds a trend for a field.
String	**getAggregate**(`String agg, String name`) Gets the value of an aggregate from the current record.
String	**getQuery**() Gets the query necessary to return the current aggregate.
int	**getTotal**(`String agg, String name`) Gets the total number of records by summing an aggregate.
String	**getValue**(`String name`) Gets the value of a field.
void	**groupBy**(`String name`) Provides the name of a field to use in grouping the aggregates. May be called numerous times to set multiple group fields.
void	**orderBy**(`String name`) Provides the name of a field that should be used to order the aggregates. The field will also be added to the group-by list.
void	**orderByAggregate**(`String agg, String name`) Provides the name of an aggregate that should be used to order the result set.
void	**query**() Issues the query and get some results.

void **setGroup**(`Boolean b`)

Sets whether grouping is true or false.

Method Detail

addEncodedQuery

public void addEncodedQuery(`String query`)

Adds a query to the Aggregate. Adds an encoded query to the other queries that may have been set for this aggregate.

Parameters:

`query` - An encoded query string to add to the aggregate.

Example:

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
     last = agg2.getAggregate('count','category');
  }
  gs.log(category + ": Last month:" + count + " Previous Month:" +
last);

}
```

addHaving

public void addHaving(String name, String operator, String value)

Adds a "having" element to the aggregate e.g. select category, count(*) from incident group by category HAVING count(*) > 5

Parameters:

`String name` - the aggregate to filter on (e.g. COUNT).

`String operator` - for the operator symbol (e.g. <, >, =, !=)

`String value` to query on (e.g. '5' or '69')

---

addAggregate

public void addAggregate(String agg, String name)

Adds an aggregate.

Parameters:

`String agg` - name of aggregate to add.

`String name` - name of column to aggregate.

Example:

```
function doMyBusinessRule(assigned_to, number) {
  var agg = new GlideAggregate('incident');
  agg.addQuery('assigned_to', assigned_to);
  agg.addQuery('category', number);
  agg.addAggregate("COUNT");
  agg.query();
  var answer = 'false';
  if (agg.next()) {
    answer = agg.getAggregate("COUNT");
    if (answer > 0)
      answer = 'true';
    else
      answer = 'false';
  }
  return answer;
}
```

---

addTrend

public void addTrend(String fieldName, String timeInterval)

Adds a trend for a field.

Parameters:

`fieldName` - The string name of the field for which trending should occur.

`timeInterval` - The time interval for the trend. The following choices are available:

- Year
- Quarter
- Date
- Week
- DayOfWeek
- Hour
- Value

getAggregate

public String getAggregate(String agg, String name)

Gets the value of an aggregate from the current record.

Parameters:

`agg` - String type of the aggregate (e.g. SUM or COUNT)

`name` - String name of the field to get aggregate from.

Returns:

`String` - the value of the aggregate

Example:

```
function doMyBusinessRule(assigned_to, number) {
  var agg = new GlideAggregate('incident');
  agg.addQuery('assigned_to', assigned_to);
  agg.addQuery('category', number);
  agg.addAggregate("COUNT");
  agg.query();
  var answer = 'false';
  if (agg.next()) {
    answer = agg.getAggregate("COUNT");
    if (answer > 0)
      answer = 'true';
    else
      answer = 'false';
  }
  return answer;
}
```

getQuery

public String getQuery()

Gets the query necessary to return the current aggregate.

Returns:

`String` - the string query.

Example:

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
     last = agg2.getAggregate('count','category');
  }
  gs.log(category + ": Last month:" + count + " Previous Month:" + 
last);

}
```

---

getTotal

public int getTotal(String agg, String name)

Gets the total number of records by summing an aggregate.

Parameters:

`agg` - String type of aggregate

`agg` - String name of field to aggregate from

Returns:

`int` - the total.

---

getValue

public String getValue(String name)

Gets the value of a field.

Parameters:

`String name` - the string name of the field.

Returns:

`String value` - the string value of the field.

groupBy

public void groupBy(String name)

Provides the name of a field to use in grouping the aggregates. May be called numerous times to set multiple group fields.

Parameters:

`name` - name of the field to group-by.

Example:

```
Referencing the example in wiki @ Aggregation Support:

  var count = new GlideAggregate('incident');
  count.addAggregate('MIN', 'sys_mod_count');
  count.addAggregate('MAX', 'sys_mod_count');
  count.addAggregate('AVG', 'sys_mod_count');
  count.groupBy('category');
  count.query();
  while (count.next()) {
     var min = count.getAggregate('MIN', 'sys_mod_count');
     var max = count.getAggregate('MAX', 'sys_mod_count');
     var avg = count.getAggregate('AVG', 'sys_mod_count');
     var category = count.category.getDisplayValue();
     gs.log(category + " Update counts: MIN = " + min + " MAX = " + max
 + " AVG = " + avg);
  }
```

orderBy

public void orderBy(String name)

Provides the name of a field that should be used to order the aggregates. The field will also be added to the group-by list.

Parameters:

`name` - name of the field used to order the aggregates.

Example:

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
     last = agg2.getAggregate('count','category');
  }
  gs.log(category + ": Last month:" + count + " Previous Month:" +
last);

}
```

orderByAggregate

public void orderByAggregate(String agg, String name)

Provides the name of an aggregate that should be used to order the result set.

Parameters:

`agg` - String type of aggregate (e.g. SUM, COUNT, MIN, MAX)

`name` - String name of field to aggregate

Example:

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
     last = agg2.getAggregate('count','category');
  }
  gs.log(category + ": Last month:" + count + " Previous Month:" +
last);

}
```

---

query

public void query()

Issues the query and gets some results.

Example:

```
var agg = new GlideAggregate('incident');
agg.addAggregate('count','category');
agg.orderByAggregate('count', 'category');
agg.orderBy('category');
agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
agg.query();
while (agg.next()) {
  var category = agg.category;
  var count = agg.getAggregate('count','category');
  var query = agg.getQuery();
  var agg2 = new GlideAggregate('incident');
  agg2.addAggregate('count','category');
  agg2.orderByAggregate('count', 'category');
  agg2.orderBy('category');
  agg2.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(3)');
  agg2.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(3)');
  agg2.addEncodedQuery(query);
  agg2.query();
  var last = "";
  while (agg2.next()) {
     last = agg2.getAggregate('count','category');
  }
  gs.log(category + ": Last month:" + count + " Previous Month:" +
last);
```

setGroup

public void setGroup(boolean b)

Sets whether grouping is true or false.

Parameters:

`b` - true if grouping is true, false if it is false.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/glide-server-apis/concept/c_GlideAggregate.html

Client Side

GlideForm (g form)

Note: *This article applies to Fuji and earlier releases. For more current information, see Glide Class Overview* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

GlideForm.js is the Javascript class used to customize forms.

- g_form is a global object in the GlideForm class.
- GlideForm.js and g_form are ***only*** used on the client (for instance, the web browser).

g_form accesses:

- Display Settings
- Field Information
- Change Field
- Change Choice List
- Form Information
- Form Action

Note: *The methods* `getControl()`, `getElement()`, *and* `getFormElement()` *are deprecated for mobile devices. For more information on using GlideForm for mobile, see Mobile Client GlideForm (g form) Scripting.*

Where To Use

These methods are used to make custom changes to the form view of records. All validation of examples was done using Client Scripts.

Some of these methods can also be used in other client scripts (such as Catalog Client Scripts or Wizard Client Scripts), but must first be tested to determine whether they will work as expected.

Enhancements

Fuji

- New methods are available for getting and setting field labels, and for adding and removing field label icons.

Display Settings

Method Summary	
Return Value	**Details**
void	**addDecoration**(`fieldName, icon, title`) Adds an icon on a field's label. This method is available starting with the Fuji release.
void	**flash**(`widgetName, color, count`) Flashes the specified color the specified number of times. Used to draw attention to a particular field.
void	**getLabelOf**(`fieldname`) Gets the plain text value of the field label. This method is available starting with the Fuji release.
void	**hideAllFieldMsgs**(`[type]`) Hides all field messages. **Optional**: Provide type to clear only "info" or "error" messages.
void	**hideErrorBox**(`input`) Hides the error message placed by showErrorBox(). ***Best Practice***: Use hideFieldMsg rather than this method whenever possible.
void	**hideFieldMsg**(`input, [clearAll]`) Hides the message placed by showFieldMsg().
void	**hideRelatedList**(`listTableName`) Hides the specified related list on the form.
void	**hideRelatedLists**() Hides all related lists on the form.
void	**removeDecoration**(`fieldName, icon, title`) Removes the icon that matches the exact same name and text. This method is available starting with the Fuji release.
void	**setDisplay**(`fieldName, boolean`) Displays the field if true. Hides the field if false. If the field is hidden, the space is reclaimed. ***Best Practice***: Use UI Policy rather than this method whenever possible.
void	**setLabelOf**(`fieldname, label`) Sets the plain text value of the field label. This method is available starting with the Fuji release.
void	**setVisible**(`fieldName, boolean`) Displays the field if true. Hides the field if false. If the field is hidden, the space is left blank. ***Best Practice***: Use UI Policy rather than this method whenever possible.
void	**showErrorBox**(`input, message, [scrollForm]`) Displays an error message under the specified form field (either a control object or the name of the field). If the control or field is currently scrolled off the screen, it will be scrolled to. **Optional**: Set scrollForm to false to prevent scrolling to the error message offscreen.
void	**showFieldMsg**(`input, message, type, [scrollForm]`) Displays either an informational or error message under the specified form field (either a control object or the name of the field). Type may be either "info" or "error." If the control or field is currently scrolled off the screen, it will be scrolled to. **Optional**: Set scrollForm to false to prevent scrolling to the field message offscreen.
void	**showRelatedList**(`listTableName`) Displays the specified related list on the form.

void	**showRelatedLists**() Displays all related lists on the form.

Field Information

Method Summary	
Return Value	**Details**
Boolean	**getBooleanValue**(`fieldName`) Returns false if the field's value is false or undefined, otherwise true is returned. Useful with checkbox fields. Returns true when the checkbox is checked.
HTMLElement	**getControl**(`fieldName`) Returns the HTML element for the specified field. Compound fields may contain several HTML elements. Generally not necessary as there are built-in methods that use the fields on the form.
HTMLElement	**getElement**(`id`) Returns the HTML element for the field specified via the ID. Compound fields may contain several HTML elements. Generally not necessary as there are built-in methods that use the fields on the form.
int	**getIntValue**(`fieldName`) Returns the value of the specified field as an integer. An empty value returns 0.
HTMLElement	**getOption**(`fieldName, choiceValue`) Returns the <option> element for a select box named fieldName and where choiceValue matches the option value. Returns null if the field is not found or the option is not found.
string	**getReference**(`fieldName, callback`) Returns the GlideRecord for a specified field. getReference() accepts a second parameter, a callback function. **Warning:** This requires a call to the server so using this function will require additional time and may introduce latency to your page.
string	**getDecimalValue**(`fieldName`) Returns the decimal value of the specified field.
string	**getValue**(`fieldName`) Returns the value of the specified field.
boolean	**isMandatory**(`fieldName`) Returns true if the field is required. Returns false if the field is optional.

Change Field

Method Summary	
Return Value	**Details**
void	**clearValue**(`fieldName`) Removes any value(s) from the specified field.
void	**setDisabled**(`fieldName, boolean`) Grays out field and makes it unavailable.
void	**setMandatory**(`fieldName, boolean`) Makes the field required if true. Makes the field optional if false. ***Best Practice***: Use UI Policy rather than this method whenever possible.
void	**setReadOnly**(`fieldName, boolean`) Makes the field read-only if true. Makes the field editable if false. **Note**: Both **setReadOnly** and **setReadonly** are functional. ***Best Practice***: Use UI Policy rather than this method whenever possible.
void	**setValue**(`fieldName, value, displayValue`) Sets the value and the display value of a field. Will display value if there is no displayValue. To improve performance by preventing a round trip, include a display value in addition to the value. *Note:* setValue can cause a stack overflow when used in an OnChange client script. This is because every time the value is set, it will register as a change, which may re-trigger the OnChange client script. To prevent this, perform a check that will validate that the new value will be different from the old value. For example, before performing `setValue(fieldName, newValue.toUpperCase());`, validate that the short description is not already uppercase. This will prevent the Client Script from applying the `toUpperCase()` more than once.

Change Choice List

Method Summary	
Return Value	**Details**
void	**addOption**(`fieldName, choiceValue, choiceLabel, [choiceIndex]`) Adds a choice to a choice list field. If the index is not specified, the choice is added to the end of the list. **Optional**: Use the index field to specify a particular place in the list.
void	**clearOptions**(`fieldName`) Removes all options from a choice list.
void	**removeOption**(`fieldName, choiceValue`) Removes a specific option from a choice list.

Form Information

Method Summary	
Return Value	**Details**
string	**getActionName**() Returns the most recent action name or, for a client script, the sys_id of the UI Action clicked. **Note:** not available to Wizard Client Scripts.
HTMLElement	**getFormElement**() Returns the HTML element for the form.
string	**getSectionNames**() Returns all section names, whether visible or not, in an array. This method is available starting with the Fuji release.
string	**getSections**() Returns the elements for the form's sections in an array.
string	**getTableName**() Returns the name of the table this record belongs to.
string	**getUniqueValue**() Returns the sys_id of the record displayed in the form.
boolean	**isNewRecord**() Returns true if the record has never been saved. Returns false if the record has been saved.
boolean	**isSectionVisible**(`sectionName`) Returns true if the section is visible. Returns false if the section is not visible or does not exist. This method is available starting with the Fuji release.

Form Action

Method Summary	
Return Value	**Details**
void	**addErrorMessage**(`message`) Displays an error message at the top of the form.
void	**addInfoMessage**(`message`) Displays an informational message at the top of the form.
void	**clearMessages**() Removes messages that were previously added with addErrorMessage() and addInfoMessage().
void	**enableAttachments**() Allows new file attachments to be added. Shows the paperclip icon. See also: disableAttachments().
void	**disableAttachments**() Prevents new file attachments from being added. Hides the paperclip icon. See also: enableAttachments().
void	**save**() Saves the record without navigating away from the record (update and stay).
boolean	**setSectionDisplay**(`sectionName, display`) Shows or hides a section. Works in both tab and flat modes. This method is available starting with the Fuji release.
void	**submit**() Saves the record. User will be taken away from the form, returning them to where they were previously.

[top of page]

Display Settings Methods

addDecoration

void addDecoration(`fieldName, icon, title`)

Adds an icon on a field’s label. Adding the same item twice is prevented; however, you can add the same icon with a different title. This method is available starting with the Fuji release.

Click the plus to view the supported icons.

- icon-user
- icon-user-group
- icon-lightbulb
- icon-home
- icon-mobile
- icon-comment
- icon-mail
- icon-locked
- icon-database
- icon-book
- icon-drawer
- icon-folder
- icon-catalog
- icon-tab
- icon-cards
- icon-tree-right
- icon-tree
- icon-book-open
- icon-paperclip
- icon-edit
- icon-trash
- icon-image
- icon-search
- icon-power
- icon-cog
- icon-star
- icon-star-empty
- icon-new-ticket
- icon-dashboard
- icon-cart-full
- icon-view
- icon-label
- icon-filter
- icon-calendar
- icon-script
- icon-add
- icon-delete
- icon-help
- icon-info

- icon-check-circle
- icon-alert
- icon-sort-ascending
- icon-console
- icon-list
- icon-form
- icon-livefeed

Parameters:

`fieldname` - The field name.

`icon` - The font icon to show next to the field.

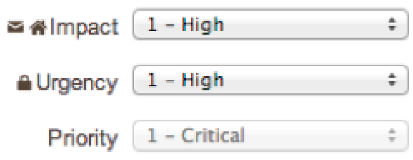
`title` - The text title for the icon.

Returns:

`void`

Example:

```
g_form.addDecoration('caller_id', 'icon-star', 'preferred member');
```



Impact	1 - High
Urgency	1 - High
Priority	1 - Critical

Note: *This method is not supported by Service Catalog.*

flash

void flash(`widgetName, color, count`)

Flashes the specified color the specified number of times in the field. Used to draw attention to a particular field.

Parameters:

`widgetName` - specifies the field with <table name>.<fieldname>.

`color` - RGB color or acceptable CSS color like "blue" or "tomato."

`count` - integer that determines how long the label will flash.

- use 2 for a 1-second flash
- use 0 for a 2-second flash
- use -2 for a 3-second flash
- use -4 for a 4-second flash

Returns:

`void`

Example:

```
g_form.flash("incident.number", "#FFFACD", 0);
```

Note: *This method is not supported by Service Catalog.*

---

getLabelOf

String getLabelOf(`fieldname`)

Gets the plain text value of the field label. This method is available starting with the Fuji release.

Parameters:

`fieldname` - The field name.

Returns:

The label text.

Example:

```
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabelOf('comments');
    g_form.setLabelOf('comments', oldLabel + ' (Customer visible)');
}
```

---

hideAllFieldMsgs

void hideAllFieldMsgs(`[type]`)

Hides all field messages.

Optional: Provide type to hide only "info" or "error" messages.

Parameters:

`type` (optional) - info or error.

Returns:

`void`

---

hideErrorBox

void hideErrorBox(`input`)

Hides the error message placed by showErrorBox().

Best Practice: Use hideFieldMsg rather than this method whenever possible.

Parameters:

`input` - specifies the name of the field or control.

Returns:

`void`

hideFieldMsg

void hideFieldMsg(`input, [clearAll]`)

Hides the message placed by showFieldMsg().

Parameters:

`input` - specifies the name of the field.

`clearAll (optional)` - boolean parameter indicating whether to clear all messages. If true, all messages for the field will be cleared; if false or empty, only the last message will be removed.

Returns:

`void`

Example:

```
g_form.hideFieldMsg('impact', true);
```

hideRelatedList

void hideRelatedList(`listTableName`)

Hides the specified related list on the form.

Parameters:

`listTableName` - specifies the name of the related list. Use the list sys_id to hide a list through a relationship.

Returns:

`void`

hideRelatedLists

void hideRelatedLists()

Hides all related lists on the form.

Returns:

`void`

removeDecoration

void removeDecoration(`fieldName, icon, title`)

Removes the icon that matches the exact same name and text. This method is available starting with the Fuji release.

Parameters:

`fieldName` - The field name..

`fieldName` - The icon to remove.

`fieldName` - The text title for the icon.

Returns:

`void`

Example:

```
function onChange(control, oldValue, newValue, isLoading) {
      // if the caller_id field is not present, then we can't add an
icon anywhere
      if (!g_form.hasField('caller_id'))
            return;

      if (!newValue)
            return;

      g_form.getReference('caller_id', function(ref) {
            g_form.removeDecoration('caller_id', 'icon-star', 'VIP');

            if (ref.getValue('vip') == 'true')
                  g_form.addDecoration('caller_id', 'icon-star',
'VIP');
      });
}
```

Note: *This method is not supported by Service Catalog.*

setDisplay

void setDisplay(`fieldName, boolean`)

Displays the field if true. Hides the field if false. This method cannot hide mandatory fields with no value.

If the field is hidden, the space is used to display other items.

Best Practice: Use UI Policy rather than this method whenever possible.

Parameters:

`fieldName` - specifies the field to be displayed or hidden.

`boolean` - true to display the field, false to hide the field.

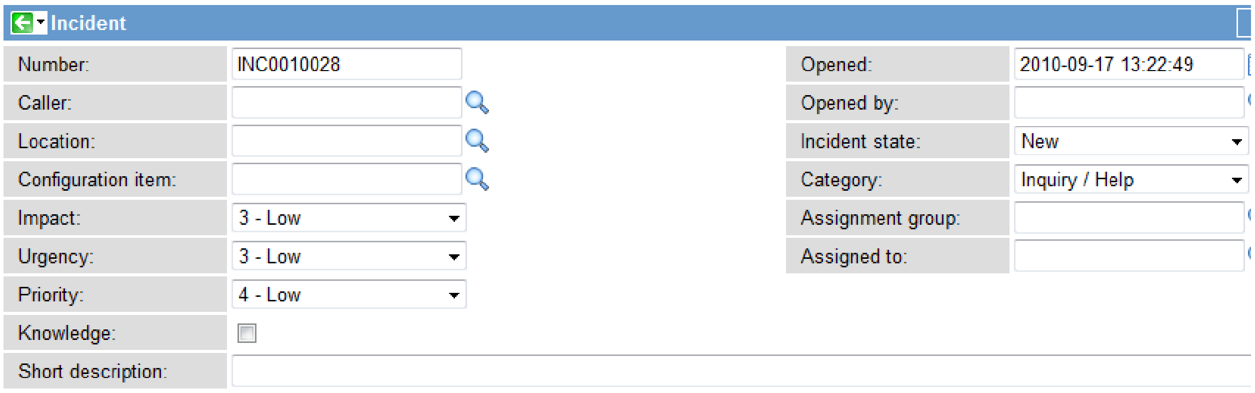
Returns:

`void`

Example:

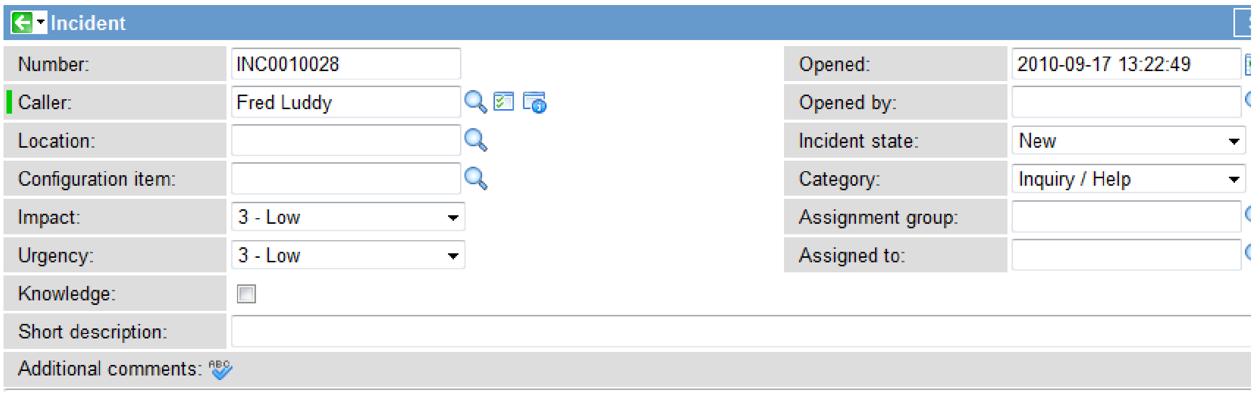
```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
   //If the page isn't loading
   if (!isLoading) {
      //If the new value isn't blank
      if (newValue != '') {
         g_form.setDisplay('priority', false);
      }
      else
         g_form.setDisplay('priority', true);
      }
   }
```

"Priority" Field Displayed



Number:	INC0010028	Opened:	2010-09-17 13:22:49
Caller:		Opened by:	
Location:		Incident state:	New
Configuration item:		Category:	Inquiry / Help
Impact:	3 - Low	Assignment group:	
Urgency:	3 - Low	Assigned to:	
Priority:	4 - Low		
Knowledge:			
Short description:			

"Priority" Field Hidden



Number:	INC0010028	Opened:	2010-09-17 13:22:49
Caller:	Fred Luddy	Opened by:	
Location:		Incident state:	New
Configuration item:		Category:	Inquiry / Help
Impact:	3 - Low	Assignment group:	
Urgency:	3 - Low	Assigned to:	
Knowledge:			
Short description:			
Additional comments:			

setLabelOf

void setLabelOf(`fieldname, label`)

Sets the plain text value of the field label. This method is available starting with the Fuji release.

Parameters:

`fieldName` - The field name.

`label` - The field label text.

Returns:

`void`

Example:

This example changes the comments label.

```
if (g_user.hasRole('itil')) {
    var oldLabel = g_form.getLabelOf('comments');
    g_form.setLabelOf('comments', oldLabel + ' (Customer visible)');
}
```

Note: *This method is not supported by Service Catalog.*

setVisible

void setVisible(`fieldName, boolean`)

Displays the field if true. Hides the field if false.

If the field is hidden, the space is left blank. This method cannot hide mandatory fields with no value.

Best Practice: Use UI Policy rather than this method whenever possible.

Parameters:

`fieldName` - specifies the field to be displayed or hidden.

`boolean` - true to display the field, false to hide the field.

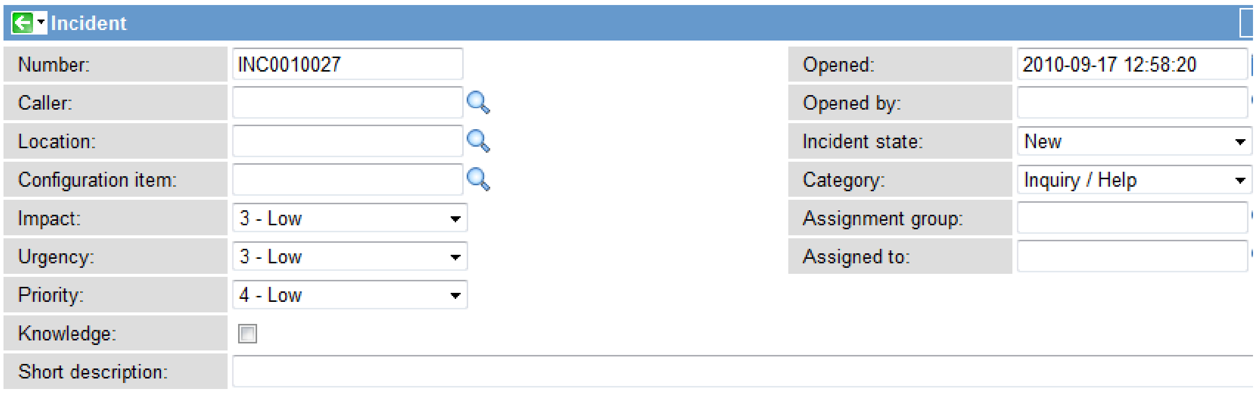
Returns:

`void`

Example:

```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
   //If the page isn't loading
   if (!isLoading) {
      //If the new value isn't blank
      if(newValue != '') {
         g_form.setVisible('priority', false);
      }
      else
         g_form.setVisible('priority', true);
      }
   }
```

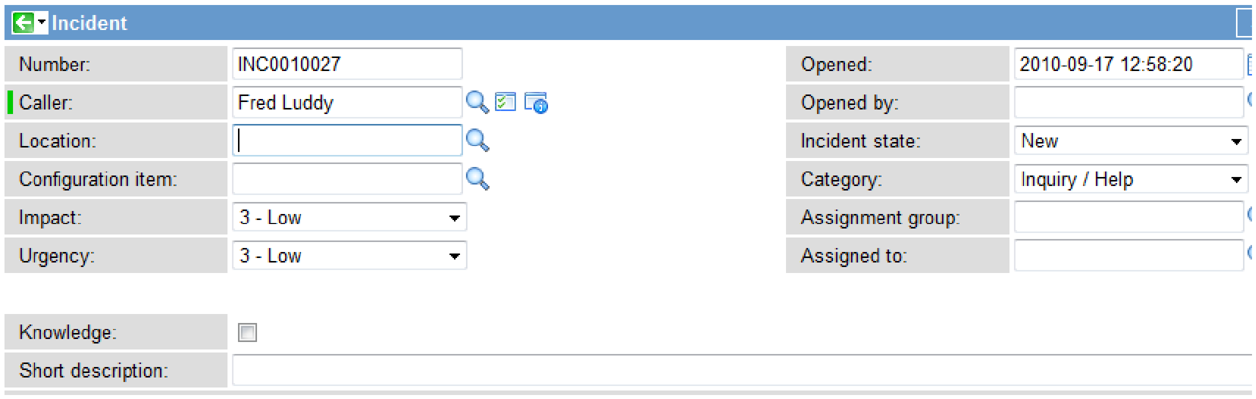
"Priority" Field Displayed



Incident

Number:	INC0010027	Opened:	2010-09-17 12:58:20
Caller:		Opened by:	
Location:		Incident state:	New
Configuration item:		Category:	Inquiry / Help
Impact:	3 - Low	Assignment group:	
Urgency:	3 - Low	Assigned to:	
Priority:	4 - Low		
Knowledge:	☐		
Short description:			

"Priority" Field Hidden



Incident

Number:	INC0010027	Opened:	2010-09-17 12:58:20
Caller:	Fred Luddy	Opened by:	
Location:		Incident state:	New
Configuration item:		Category:	Inquiry / Help
Impact:	3 - Low	Assignment group:	
Urgency:	3 - Low	Assigned to:	
Knowledge:	☐		
Short description:			

showErrorBox

void showErrorBox(`input, message, [scrollForm]`)

Displays an error message under the specified form field (either a control object or the name of the field). If the control or field is currently scrolled off the screen, it will be scrolled to.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is offscreen (scrolls the form to the control or field).

The showFieldMsg() method is a similar method that requires a "type" parameter.

Optional: Set scrollForm to false to prevent scrolling to the error message offscreen.

Parameters:

`input` - specifies the name of the field or control.

`message` - the message to be displayed.

`scrollForm` (optional) - true to scroll to message if offscreen, false to prevent this scrolling.

Returns:

`void`

Note: The scroll form functionality for `showErrorBox` does not work in CMS with Catalog Items.

showFieldMsg

void showFieldMsg(`input, message, type, [scrollForm]`)

Displays either an informational or error message under the specified form field (either a control object or the name of the field). Type may be either "info" or "error." If the control or field is currently scrolled off the screen, it will be scrolled to.

A global property (glide.ui.scroll_to_message_field) is available that controls automatic message scrolling when the form field is offscreen (scrolls the form to the control or field).

The showErrorBox() method is a shorthand method that does not require the "type" parameter.

Optional: Set scrollForm to false to prevent scrolling to the field message offscreen.

Parameters:

`input` - specifies the name of the field or control.

`message` - the message to be displayed.

`type` - error or info.

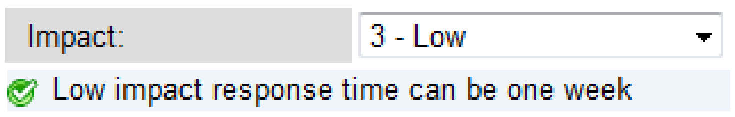
`scrollForm` (optional) - true to scroll to message if offscreen, false to prevent this scrolling.

Returns:

`void`

Example - Informational Message:

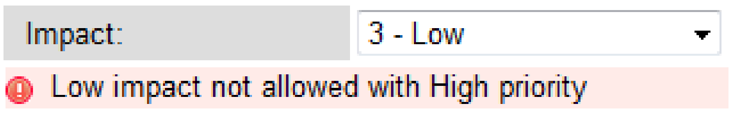
```
g_form.showFieldMsg('impact','Low impact response time can be one
week','info');
```



Low impact response time can be one week

Example - Error Message:

```
g_form.showFieldMsg('impact','Low impact not allowed with High
priority','error');
```


Impact:
3 - Low
Low impact not allowed with High priority

showRelatedList

void showRelatedList(`listTableName`)

Displays the specified related list on the form.

Parameters:

`listTableName` - specifies the name of the related list.

Returns:

`void`

showRelatedLists

void showRelatedLists()

Displays all related lists on the form.

Returns:

`void`

[top of page]

Field Information Methods

getBooleanValue

Boolean getBooleanValue(`fieldName`)

Returns false if the field value is false or undefined, otherwise returns true.

Parameters:

`fieldName` - specifies the name of the field.

Returns:

`Boolean` - the boolean value of the field.

getControl

HTMLElement getControl(`fieldName`)

Returns the HTML element for the specified field. Compound fields may contain several HTML elements.

Generally not necessary as there are built-in methods that use the fields on the form.

Parameters:

`fieldName` - specifies the name of the field.

Returns:

`HTMLElement` - the specified HTML element(s).

Note: If the field is of type reference and the control is a choice list, getControl() may not return control as expected. In this case, use `sys_select.<table name>.<field name>`.

---

getElement

HTMLElement getElement(`id`)

Returns the HTML element for the field specified by the ID. Compound fields may contain several HTML elements.

Generally not necessary as there are built-in methods that use the fields on the form.

Parameters:

`id` - specifies the field ID.

Returns:

`HTMLElement` - the specified HTML element(s).

---

getIntValue

int getIntValue(`fieldName`)

Returns the value of the specified field as an integer. An empty value returns 0.

Closely related to **getValue().**

Parameters:

`fieldName` - specifies the field.

Returns:

`integer` - value of the specified field as an integer.

---

getOption

HTMLElement getOption(fieldName, choiceValue)

Returns the <option> element for a select box named fieldName and where choiceValue matches the option value.

Returns null if the field is not found or the option is not found.

Parameters:

`fieldName` - specifies the name of the field.

`choiceValue` - specifies the value of the option to return.

Returns:

HTMLElement - the specified HTML element.

getReference

GlideRecord getReference(`fieldName`, `callback`)

Returns the GlideRecord for the record specified in a reference field. If the referenced record cannot be found, then it returns an initialized GlideRecord object where currentRow = -1 and rows.length = 0.

getReference() accepts a second parameter, a callback function.

Callback function support for `ServiceCatalogForm.getReference` is available.

Warning: This requires a call to the server so using this function will require additional time and may introduce latency to your page. Use with caution. See Avoid Server Lookups.

Parameters:

`fieldName` - specifies the reference field.

Returns:

`GlideRecord` - the GlideRecord object for the record in specified in the reference field. See GlideRecord for more information.

- If a callback function is present, this routine runs asynchronously, and browser (and script) processing will continue normally until the server returns the reference value, at which time the callback function will be invoked.
- If a callback function is not present, this routine runs synchronously and processing will halt (causing the browser to appear to hang) while waiting on a server response. It is strongly recommended that a callback function be used on any supported versions.

Example: without a callback (don't do this)

```
function onChange(control, oldValue, newValue, isLoading) {
   var caller = g_form.getReference('caller_id');
   if (caller.vip == 'true')
      alert('Caller is a VIP!');
}
```

Example: with a callback (recommended)

```
function onChange(control, oldValue, newValue, isLoading) {
   var caller = g_form.getReference('caller_id', doAlert); // doAlert
is our callback function
}
function doAlert(caller) { //reference is passed into callback as first
 arguments
  if (caller.vip == 'true')
    alert('Caller is a VIP!');
}
```

---

getValue

string getValue(fieldName)

Returns the value of the specified field.

Parameters:

fieldName - specifies the field.

Returns:

string - the value of the specified field.

Example:

```
function onChange(control, oldValue, newValue, isLoading) {
   alert(g_form.getValue('short_description'));
}
```

Example:

```
  function onChange(control, oldValue, newValue, isLoading)
  if (isLoading)
      return;

  if (!g_form.isNewRecord())
      return;

  var lookup = new GlideRecord('short_desc_lookup');
  lookup.addQuery('u_subcategory', g_form.getValue('subcategory'));
  lookup.query();
  var temp;  //temp var - reusable
    if (lookup.next()) {
    temp = lookup.u_short_description;
      if (null != temp){    //Set the form value from lookup if there is
 a lookup value
      g_form.setValue('short_description', temp);
      }else{
      g_form.setValue('short_description', "");
      }
    }else{
      //If a lookup record does not exist based on lookup.addQuery
      //Then set to UNDEFINED or NULL depending on type
    g_form.setValue('short_description', "");
    }
  }

}

----
function onLoad() {
   if(g_form.isNewRecord()){
      alert('New Record!');
   }
}
```

getDecimalValue

string getDecimalValue(`fieldName`)

Returns the decimal value of the specified field.

Parameters:

`fieldName` - specifies the field.

Returns:

`string` - the value of the specified field.

Example:

```
function onChange(control, oldValue, newValue, isLoading) {
   alert(g_form.getValue('percent_complete'));
}
```

isMandatory

boolean isMandatory(`fieldName`)

Returns true if the field is mandatory.

Parameters:

`fieldName` - specifies the field.

Returns:

`boolean` - true if the field is required, false if it is optional.

[top of page]

Change Field Methods

clearValue

void clearValue(`fieldName`)

Removes any value(s) from the specified field.

Parameters:

`fieldName` - specifies the field.

Returns:

`void`

setDisabled

void setDisabled(`fieldName, boolean`)

Grays out field and makes it unavailable.

Parameters:

`fieldName` - specifies the field.

`boolean` - true if disabled, false if enabled.

Returns:

`void`

---

setMandatory

void setMandatory(`fieldName, boolean`)

Makes the field required if true.

Best Practice: Use UI Policy rather than this method whenever possible.

Parameters:

`fieldName` - specifies the field.

`boolean` - true if mandatory, false if optional.

Returns:

`void`

---

setReadOnly

void setReadOnly(`fieldName, boolean`)

Makes the field read-only if true, editable if false. To make a mandatory field read-only, you must first remove the mandatory requirement for that field by using the **setMandatory** method.

Note: Both **setReadOnly** and **setReadonly** are functional**.**

Best Practice: Use UI Policy rather than this method whenever possible.

Parameters:

`fieldName` - specifies the field.

`boolean` - true if read-only, false if editable.

Returns:

`void`

---

setValue

void setValue(`fieldName, value, [displayValue]`)

Sets the value and the display value (if provided) of a field. When defining a value in a choice list, be sure to use number value rather than the label.

To improve performance by preventing a round trip, include a display value in addition to the value.

Note: setValue can cause a stack overflow when used in an OnChange client script. This is because every time the value is set, it will register as a change, which may re-trigger the OnChange client script. To prevent this, perform a check that will validate that the new value will be different from the old value. For example, before performing `setValue(fieldName, newValue.toUpperCase());`, validate that the short description is not already uppercase. This will prevent the Client Script from applying the `toUpperCase()` more than once.

Parameters:

`fieldName` - specifies the field.

`value` - value in database.

`displayValue` (optional) - value that will be displayed.

Returns:

`void`

Example: Setting Display to Value

```
g_form.setValue('short_description', 'replace this with appropriate
text');
```

Example: Setting Display to Display Values

```
var valueArray = new Array("46d44a23a9fe19810012d100cca80666",
"46c6f9efa9fe198101ddf5eed9adf6e7");
var labelArray = new Array("Beth Anglin", "Bud Richman");
g_form.setValue("watch_list", valueArray, labelArray);
```

Example: Setting Display to Glide List

```
//Set Assignment Group to CI's support group if assignment group is
empty
function onChange(control, oldValue, newValue, isLoading) {

   if (!isLoading) {
      if (newValue) {
         if (newValue != oldValue) {
            if (g_form.getValue('assignment_group' != '') {
               var ciSupportGroup =
g_form.getReference('cmdb_ci').support_group;
               if (ciSupportGroup != '')
                  g_form.setValue('assignment_group', ciSupportGroup);
            }
         }
      }
   }
}
```

[top of page]

Change Choice List Methods

addOption

void addOption(`fieldName, choiceValue, choiceLabel, [choiceIndex]))`

Adds a choice to a choice list field. If the index is not specified, the choice parameter is added to the end of the list.

Optional: Use the index field to specify a particular place in the list.

Parameters:

`fieldName` - specifies the field.

`choiceValue` - the value stored in the database.

`choiceLabel` - the value displayed.

`choiceIndex` (optional) - order of choice in the list.

Returns:

`void`

Example - Add an Option to the End of a Choice List:

```
g_form.addOption('priority', '6', '6 - Really Low');
```

Example - Add an Option at a Specific Point on a Choice List:

```
g_form.addOption('priority', '2.5', '2.5 - Moderately High', 3);
```

Note that an insert will happen at the location you specify on a **zero based array**. Existing options will all shift up one level. For example, given:

[A, B, C, D]

Insert E at 0

[**E**, A, B, C, D]

Insert E at 2

[A, B, **E**, C, D]

---

clearOptions

void clearOptions(`fieldName`)

Removes all options from a choice list.

Parameters:

`fieldName` - specifies the field.

Returns:

`void`

---

removeOption

void removeOption(`fieldName, choiceValue`)

Removes a specific option from a choice list.

Note: (Versions prior to Eureka) When items are removed from a choice list by a Client Script, Google Chrome and Apple Safari will still display those items. When one of these web browsers is used, the "removed" items will be displayed as read only in the choice list and they cannot be chosen.

Parameters:

`fieldName` - specifies the field.

`choiceValue` - specifies the value stored in the database as a choice.

Returns:

`void`

Example: Remove the '1' Option from the Priority Field

```
g_form.removeOption('priority', '1');
```

Example: Remove the 'Closed' State Option if the User is not an Admin

```
function onLoad() {
   var isAdmin = g_user.hasRole('admin');
   var incidentState = g_form.getValue('incident_state');
   if (!isAdmin && (incidentState != 7)){
      g_form.removeOption('incident_state', '7');
   }
}
```

[top of page]

Form Information Methods

getActionName

string getActionName()

Returns the most recent action name or, for a client script, the sys_id of the UI Action clicked. **Note:** not available in Wizard Client Scripts.

Returns:

`string` - current action name.

Example:

```
function onSubmit() {
   var action = g_form.getActionName();
   alert('You pressed ' + action);
}
```

getFormElement

HTMLFormElement getFormElement()

Returns the HTML element for the form.

Returns:

`HTMLFormElement` - returns the HTML element for the form.

getSectionNames

array getSectionNames()

Returns all section names, whether visible or not, in an array.

Returns:

`array` - the segment names.

getSections

array getSections()

Returns the elements for the form's sections in an array.

Returns:

`array` - the HTML Elements for the form.

Example:

```
function onChange(control, oldValue, newValue, isLoading) {
   //this example was run on a form divided into sections (Change form)
   // and hid a section when the "state" field was changed
   var sections = g_form.getSections();
   if (newValue == '2') {
      g_form.setSectionDisplay(sections[1], false);
   } else {
      g_form.setSectionDisplay(sections[1], true);
   }
}
```

getTableName

string getTableName()

Returns the name of the table this record belongs to. On the server side, the table for the current record can be retrieved with `current.sys_class_name` or current.getTableName().

Returns:

`string` - table name.

Example:

```
function onLoad() {
    if (g_form.isNewRecord() {
        var tableName = g_form.getTableName(); //Get the table name
    }
}
```

getUniqueValue

string getUniqueValue()

Returns the sys_id of the record displayed in the form.

Returns:

`string` - sys_id.

Example:

```
function onLoad() {
   var incSysid = g_form.getUniqueValue();
   alert(incSysid);
}
```

---

isNewRecord

boolean isNewRecord()

Returns true if the record has never been saved.

Returns:

`boolean` - true if record has not been saved, false if it has been saved.

Example:

```
function onLoad() {
   if(g_form.isNewRecord()){
      alert('New Record!');
   }
}
```

Example:

```
function submitConfirm() {
    if (confirm('Are you sure you want to submit a priority one
incident?')) {
        g_form.setValue('priority', '1');
        if (g_form.isNewRecord()) {
            g_form.save();
        } else {
            g_form.submit();
        }
    }
}
```

---

isSectionVisible

boolean isSectionVisible(`sectionName`)

Returns true if the section is visible. Returns false if the section is not visible.

Returns:

`boolean` - true if the section is visible. Returns false if the section is not visible.

[top of page]

Form Action Methods

addErrorMessage

void addErrorMessage(`message`)

Displays an error message at the top of the form.

Parameters:

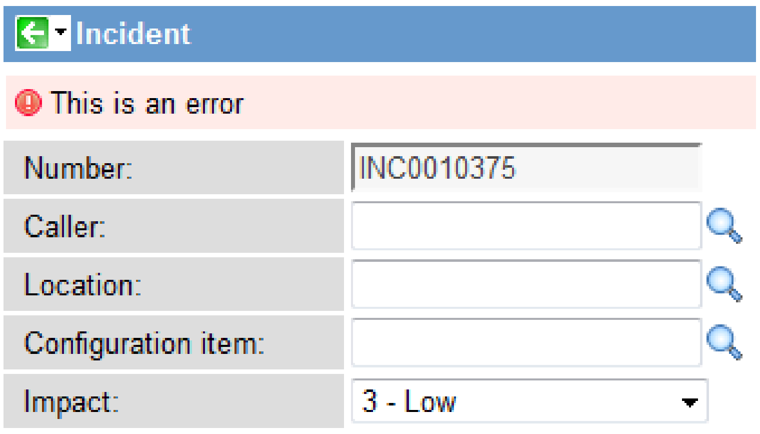
`message` - the error message to be displayed.

Returns:

`void`

Example

```
g_form.addErrorMessage('This is an error');
```


Incident

This is an error

Number:	INC0010375
Caller:	
Location:	
Configuration item:	
Impact:	3 - Low

addInfoMessage

void addInfoMessage(`message`)

Displays an informational message at the top of the form.

Parameters:

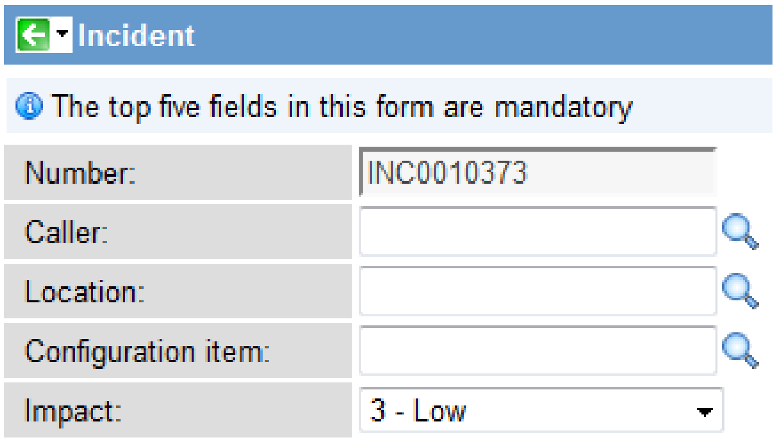
`message` - the informational message to be displayed.

Returns:

`void`

Example

```
g_form.addInfoMessage('The top five fields in this form are mandatory');
```


Incident

The top five fields in this form are mandatory

Number:	INC0010373
Caller:	
Location:	
Configuration item:	
Impact:	3 - Low

clearMessages

void clearMessages()

Removes all informational and error messages that were added with both g_form.addInfoMessage and g_form.addErrorMessage.

Returns:

`void`

Example

```
g_form.clearMessages();
```

enableAttachments

void enableAttachments()

Allows new file attachments to be added. Shows the paperclip icon. See also: disableAttachments().

Returns:

`void`

disableAttachments

void disableAttachments()

Prevents new file attachments from being added. Hides the paperclip icon.

Returns:

`void`

save

void save()

Saves the record without navigating away from the record (update and stay).

Returns:

`void`

setSectionDisplay

boolean setSectionDisplay(`sectionName, display`)

Parameters:

`sectionName` - name of the section to be shown or hidden. The section name is lower case with an underscore replacing the first space in the name, and with the remaining spaces being removed, for example "Section Four is Here" becomes "section_fourishere". Other non-alphanumeric characters, such as ampersand (&), are removed. Section names can be found by using the getSectionNames method.

`display` - set true to show, false to hide.

Returns:

`Boolean` - true if successful.

submit

void submit()

Saves the record. User will be taken away from the form, returning them to where they were previously.

Returns:

```
void
```

[top of page]

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/general-scripting/reference/r_GlideClassOverview.html

GlideUser (g user)

Note: *This article applies to Fuji and earlier releases. For more current information, see Glide Class Overview* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

g_user is a global object in GlideUser (the Javascript class).

g_user

- can ***only*** be used in Client Scripts.
- contains name and role information about the current user.
- is typically used in Client Scripts and UI Policies but is also found in UI Actions which run on the client.
- cannot be used in Business Rules or UI Actions which run on the server.
- avoids the need for much slower user queries (for example, GlideRecord queries to get user information)

Session information about the current user and current user roles is contained in the client (for example, web browser). All methods below [except getClientData()] access the session information that is available by default. getClientData() requires setup on the server and use of putClientData() to make session information available.

Property Summary	
Property	**Description**
userName	User name of the current user.
userID	Sys_id of the current user.
firstName	First name of the current user.
lastName	Last name of the current user.

Method Summary	
Return Value	**Details**
string	**getClientData**(`key`) Gets information for use in client scripts ***without*** making an AJAX call to the server. Works with putClientData(). For more information, see Session Client Data.
string	**getFullName**() Returns the first and last name of the current user.
boolean	**hasRole**(`role`) Returns true if the current user has the specified role ***or*** the admin role. **Note:** this method only accepts one role. For multiple, use hasRoles(),
boolean	**hasRoleExactly**(`role`) Returns true ***only*** if the current user has this specified role. **Warning**: Use with caution as other methods return true if the user has the specified role ***or*** the admin role. This one does not.
boolean	**hasRoleFromList**(`roles`) Returns true if the current user has at least one of the specified roles in the list ***or*** the admin role.
boolean	**hasRoles**() Returns true if the current user has ***any*** role.

[top of page]

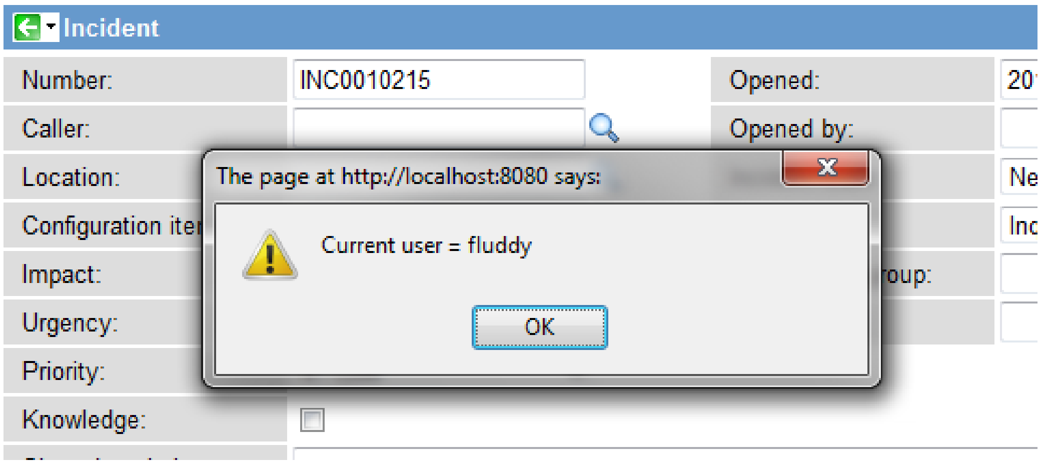
Properties

userName

User name of the current user (for example, "fluddy").

Example:

```
function onLoad() {
   var userName = g_user.userName;
   alert('Current user = ' + userName);
}
```



The page at http://localhost:8080 says:

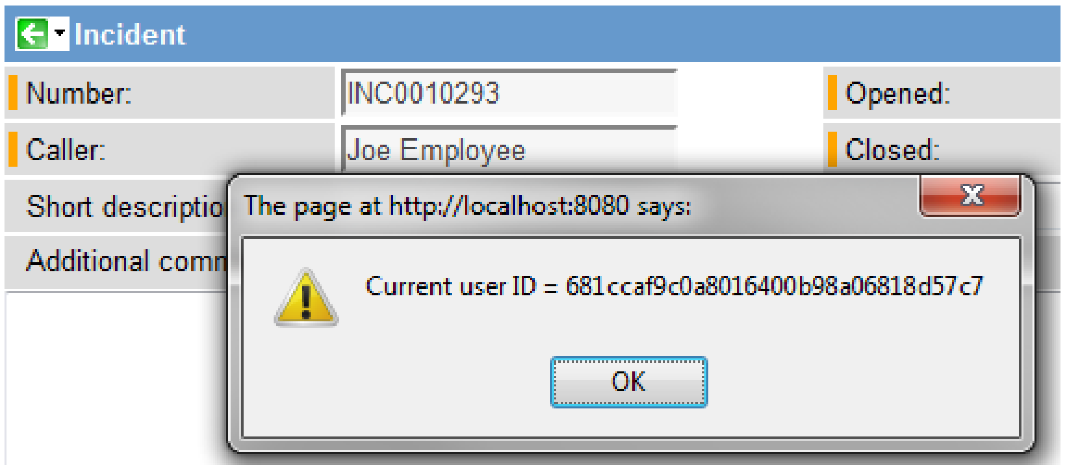
Current user = fluddy

userID

Sys_id of the current user (for example, "681ccaf9c0a8016400b98a06818d57c7").

Example:

```
function onLoad() {
   var userID = g_user.userID;
   alert('Current user ID = ' + userID);
}
```



The page at http://localhost:8080 says:

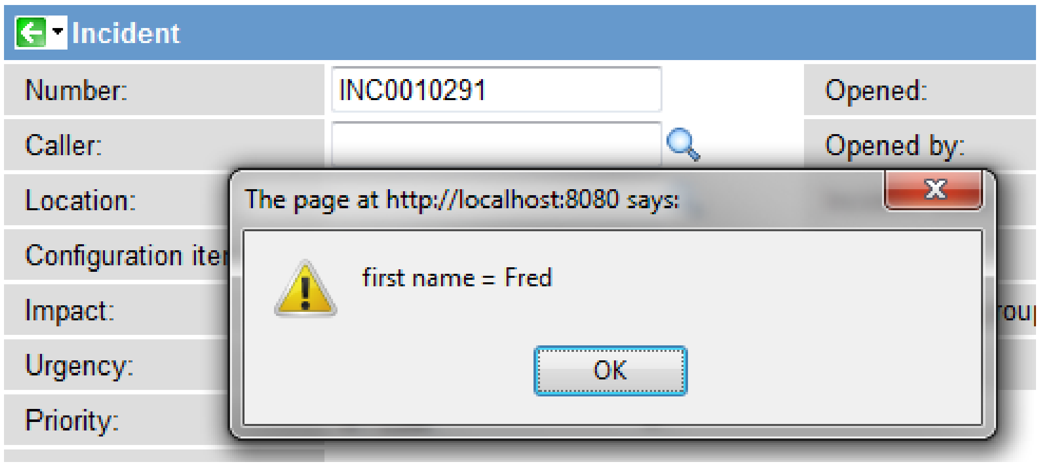
Current user ID = 681ccaf9c0a8016400b98a06818d57c7

firstName

First name of the current user (for example, "Fred"). See also **getFullName**().

Example:

```
function onLoad() {
   alert('first name = ' + g_user.firstName);
}
```



The page at http://localhost:8080 says:

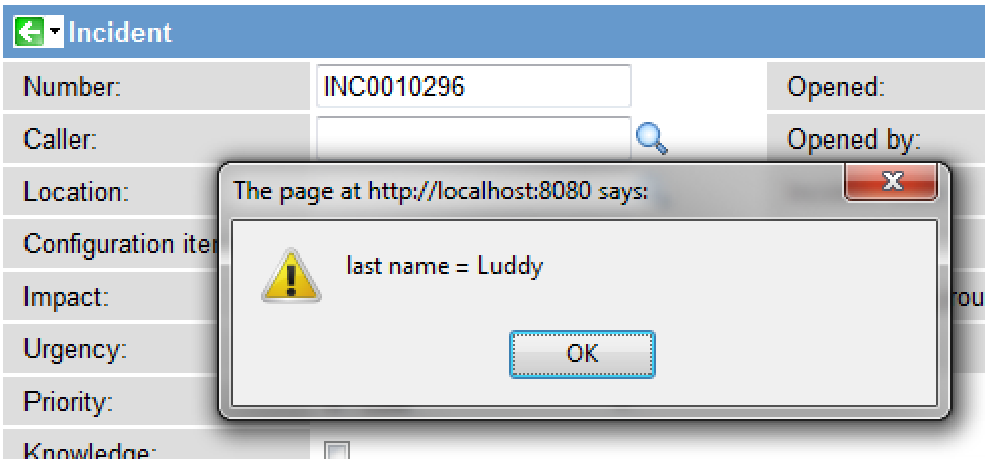
first name = Fred

lastName

Last name of the current user (for example, "Luddy"). See also **getFullName()**.

Example:

```
function onLoad() {
   alert('last name = ' + g_user.lastName);
}
```



last name = Luddy

[top of page]

Methods

getClientData

string getClientData(`key`)

Gets information for use in client scripts ***without*** making an AJAX call to the server. Works with putClientData().

Session Client Data is a set of named strings that may be setup on the server (using putClientData) that then may be used by client scripts (using getClientData). For more information see Session Client Data.

Can be used during form load time to get information that the client script needs to make decisions about the form (for example, which fields should be visible).

Parameters:

`key` - a string containing an arbitrary value used to connect putClientData and getClientData.

Returns:

`string` - contains current session information.

Example:

This example has three parts:

- Script Action - makes session data available
- UI Action - adds a button to the incident form
- Client Script - adds an alert when an incident form loads

Example Part 1 - Script Action

- When a user logs in, the login language is made available to client scripts (via getClientData)
- Event name: session.established
- Synchronous must be checked - this option may need to be added to the form and can only be added by a user with the "maint" role

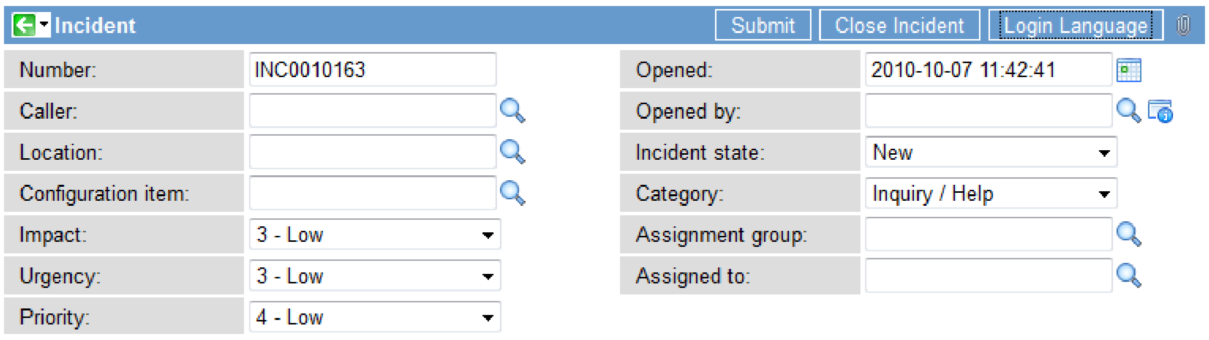
```
myLoginAction();

function myLoginAction() {
   var loginLanguage = gs.getSession().getLanguage()
   gs.getSession().putClientData("loginlanguage", loginLanguage);
}
```

Example Part 2 - UI Action

- Adds a Login Language button on the incident form that triggers the showLoginLanguage function
- Action name: login_lang_btn
- Select "Form Button" and "Client"
- Onclick: showLoginLanguage()

```
function showLoginLanguage(){
   var loginLanguage = g_user.getClientData("loginlanguage");
   if (typeof loginLanguage == "undefined")
      loginLanguage = "(we're not sure yet, check back soon)";
   alert("You logged in with " + loginLanguage);
}
```

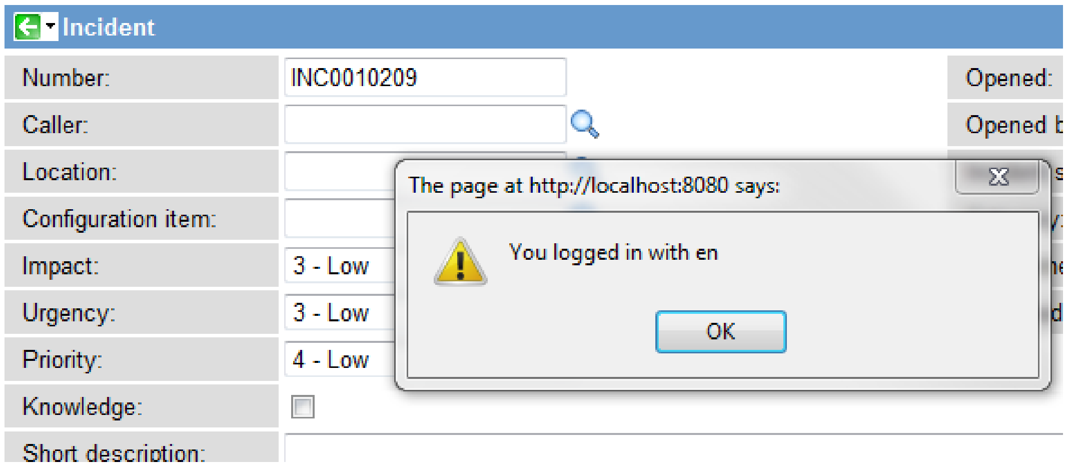


Number:	INC0010163	Opened:	2010-10-07 11:42:41
Caller:		Opened by:	
Location:		Incident state:	New
Configuration item:		Category:	Inquiry / Help
Impact:	3 - Low	Assignment group:	
Urgency:	3 - Low	Assigned to:	
Priority:	4 - Low		

Example Part 3 - Client Script

- Displays an alert when loading an incident form

```
function onLoad(){
   var loginLanguage = g_user.getClientData("loginlanguage");
   if (typeof loginLanguage == "undefined")
      loginLanguage = "(we're not sure yet, check back soon)";
   alert("You logged in with " + loginLanguage);
}
```



Number:	INC0010209	Opened:
Caller:		Opened b
Location:		
Configuration item:		
Impact:	3 - Low	
Urgency:	3 - Low	
Priority:	4 - Low	
Knowledge:	☐	
Short description:		

The page at http://localhost:8080 says:

You logged in with en

OK

---

getFullName

string getFullName()

Returns the first and last name of the current user.

Returns:

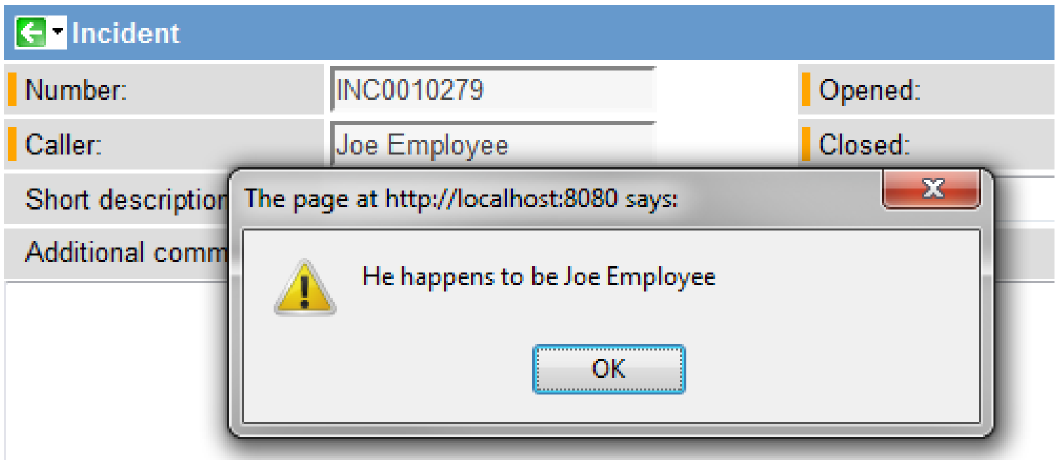
`string`

Returns:

`boolean` - returns true if the current user has the selected role or the admin role.

Example:

```
function onLoad() {
   var formalName = g_user.getFullName();
   alert ('He happens to be ' + formalName);
}
```



Number:	INC0010279	Opened:
Caller:	Joe Employee	Closed:
Short descriptio		
Additional comm		

The page at http://localhost:8080 says:

He happens to be Joe Employee

OK

hasRole

boolean hasRole(`role`)

Returns true if the current user has the selected role ***or*** the admin role.

Note: this method only accepts one role. For multiple, use hasRoleFromList(),

Parameters:

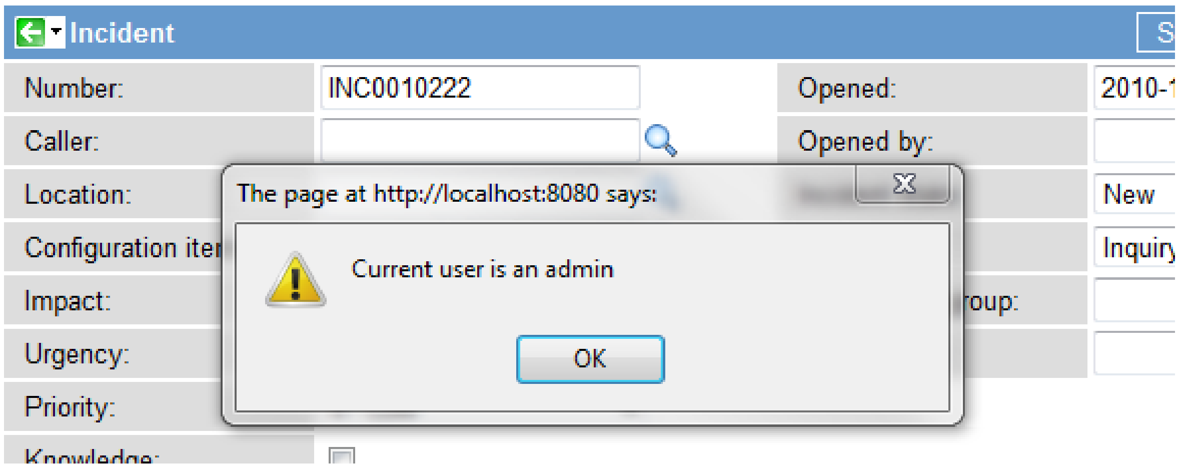
`role` - specifies the role.

Returns:

`boolean` - returns true if the current user has the selected role or the admin role.

Example:

```
function onLoad(){
   var isAdmin = g_user.hasRole('admin');
   if (isAdmin)
      alert('Current user is an admin');
   else
      alert('Current user is NOT an admin');
}
```


The page at http://localhost:8080 says:

Current user is an admin

hasRoleExactly

boolean hasRoleExactly(`role`)

Returns true ***only*** if the current user has this specified role.

Warning: Use with caution as other methods return true if the user has the specified role ***or*** the admin role. This one does not.

Parameters:

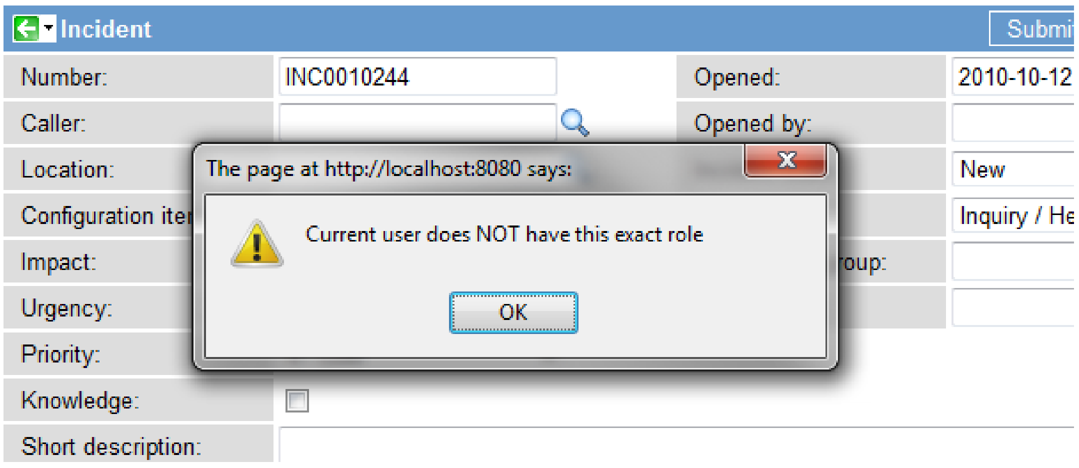
`role` - specifies the role.

Returns:

`boolean` - returns true if the current user has this role.

Example:

```
function onLoad() {
  var isItil = g_user.hasRoleExactly('itil');
  if (isItil)
    alert('Current user has this exact role');
  else
    alert('Current user does NOT have this exact role');
}
```



Current user does NOT have this exact role

---

hasRoleFromList

boolean hasRoleFromList(`roles`)

Returns true if the current user has at least one of the specified roles in the list ***or*** the admin role.

Parameters:

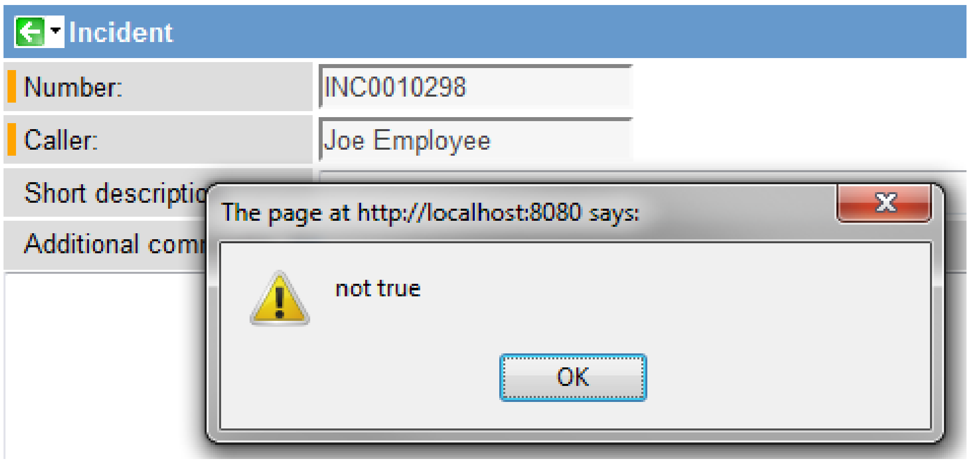
`roles` - comma-separated list of roles.

Returns:

`boolean` - returns true if the current user has a role in this list or the admin role.

Example:

```
function onLoad() {
   var isOK = g_user.hasRoleFromList("itil, maint");
   if (isOK)
      alert('true');
   else
      alert('not true');
}
```


Incident
Number: INC0010298
Caller: Joe Employee
Short descriptio
Additional comm
The page at http://localhost:8080 says:
not true
OK

hasRoles

boolean hasRoles()

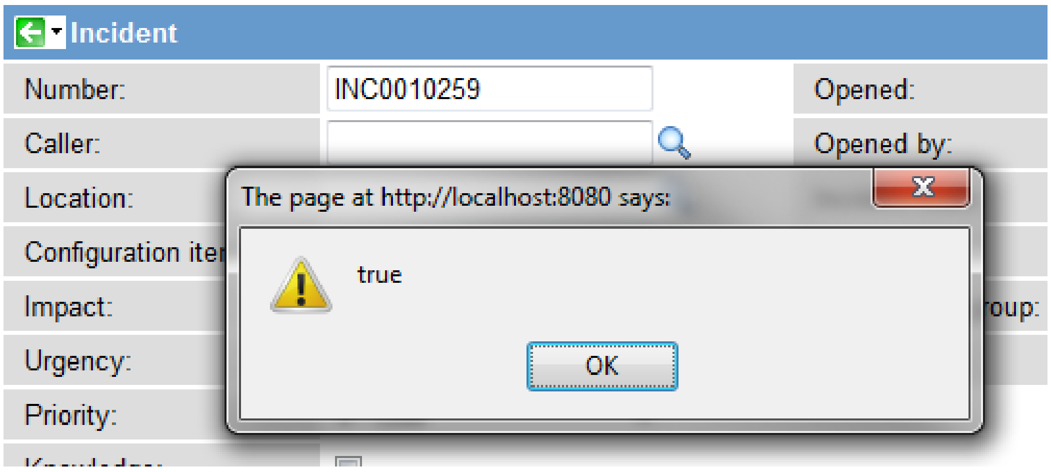
Returns true if the current user has ***any*** role.

Returns:

`boolean` - true if the current user has at least one role.

Example:

```
function onLoad() {
   var yesRole = g_user.hasRoles();
   if (yesRole)
      alert('true');
   else
      alert('not true');
}
```


Incident
Number: INC0010259
Caller:
Location:
Configuration ite
Impact:
Urgency:
Priority:
Opened:
Opened by:
oup:
The page at http://localhost:8080 says:
true
OK

[top of page]

Script Helper Objects

CIUtils

Overview

CIUtils is a utility class for working with configuration items.

Where To Use

These methods are available to any server-side script.

Information

- By default, when traversing CI relationships the system will use a max depth of 10. This can be overridden in the **glide.relationship.max_depth** property.
- The maximum number of items returns is 1000. This can be overridden in the **glide.relationship.threshold** property.

Method Summary	
Return Object	**Details**

Array

servicesAffectedByCI (`String id`)

Determine which business services are affected by a specific CI

Parameters:

`String id` - sys_id of a configuration item (cmdb_ci)

Returns:

`Array` - an array of sys_id values for cmdb_ci_server records downstream of (or affected by) the input item

Example:

```
var CIUtil = new CIUtils();

//get a server record
var server = new GlideRecord("cmdb_ci_server");
server.addQuery("name", "lnux100");
server.query();
if (server.next()) {
  //get the affected services, array of ids
  var serviceIds = CIUtil.servicesAffectedByCI(server.getUniqueValue());
  for (var i=0; i < serviceIds.length; i++) {
    //get the service record
    var service = new GlideRecord("cmdb_ci_service");
    service.get(serviceIds[i]);
    gs.print(service.getDisplayValue());
  }
}
```

```
*** Script: Client Services
*** Script: IT Services
*** Script: Bond Trading
```

Array	**servicesAffectedByTask (`GlideRecord`)**

Determine which business services are affected by a task

Parameters:

`GlideRecord` - a task GlideRecord (e.g., incident, change_request, problem)

Returns:

`Array` - an array of sys_id values for CIs downstream of (or affected by) the configuration item referenced by the task's cmdb_ci field

Example:

```
var CIUtil = new CIUtils();

//get an incident record
var inc = new GlideRecord("incident");
inc.addQuery("number", "INC00050");
inc.query();
if (inc.next()) {
  //get the affected services, array of ids
  var serviceIds = CIUtil.servicesAffectedByTask(inc);
  for (var i=0; i < serviceIds.length; i++) {
    //get the service record
    var service = new GlideRecord("cmdb_ci_service");
    service.get(serviceIds[i]);
    gs.print(service.getDisplayValue());
  }
}
```

```
*** Script: IT Services
*** Script: Email
*** Script: Windows Mobile
*** Script: Electronic Messaging
*** Script: Outlook Web Access (OWA)
*** Script: Blackberry
```

ArrayUtil

Overview

ArrayUtil is a script include with some functions that are useful when working with Javascript arrays.

Where To Use

These methods are available to any server-side script.

Method Summary

Return Object	Details

int	**contains (`Array array, Object element`)**
	searches the array for the element, returns true if the element exists in the array, otherwise returns false
	Parameters:
	`Array array` - Array to search
	`Object element` - element to search for
	Returns:
	`int` -
	Example:
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c");`
	
	`gs.print("Contains b: " + arrayUtil.contains(a1, "b"));`
	`gs.print("Contains x: " + arrayUtil.contains(a1, "x"));`
	
	`*** Script: Contains b: true`
	`*** Script: Contains x: false`

int	**indexOf (`Array array, [int startIndex]`)**
	
	searches the array for the element
	
	Parameters:
	`Array array` - Array to search
	`[int startIndex]` - optional element index to start the search
	Returns:
	`int` - returns the element position if the item exists in the array, otherwise returns -1
	Example:
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c");`
	
	`gs.print("indexOf b: " + arrayUtil.indexOf(a1, "b"));`
	`gs.print("indexOf x: " + arrayUtil.indexOf(a1, "x"));`
	
	`*** Script: indexOf b: 1`
	`*** Script: indexOf x: -1`

Array	**ensureArray (`Object`)** **returns an array from object** **Parameters:** `Object` - **Returns:** `Array` -

Array	**concat (`Array parent, Array child`)**
	
	merge two arrays
	
	Parameters:
	`Array parent` -
	`Array child` -
	Returns:
	`Array` - Array of elements from both input arrays, duplicates are not removed
	Example:
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c");`
	`var a2 = new Array("c", "d", "e");`
	
	`gs.print("concat a1, a2: " + arrayUtil.concat(a1, a2));`
	
	`*** Script: concat a1, a2: a,b,c,c,d,e`

Array	**convertArray (`Object a`)**
	
	convert an object to an array
	
	Parameters:
	
	`Object a -`
	
	Returns:
	
	`Array -`

Array	**diff (`Array a, Array b, Array (n number of arrays can be supplied)`)**
	
	find the differences between two or more arrays
	
	Parameters:
	
	`Array a -`
	
	`Array b -`
	
	`Array (n number of arrays can be supplied) -`
	
	Returns:
	
	`Array` - returns an array of items from array a that were not found in either b, c, or other input arrays, duplicates are removed from the result
	
	Example:
	
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c");`
	`var a2 = new Array("c", "d", "e");`
	`gs.print(arrayUtil.diff(a1, a2));`
	
	`*** Script: a,b`

Array	**intersect (`Array a, Array b, Array (n number of arrays can be supplied)`)**
	
	find the intesect between two or more arrays
	
	Parameters:
	
	`Array a -`
	
	`Array b -`
	
	`Array (n number of arrays can be supplied) -`
	
	Returns:
	
	`Array` - returns an array of items from array a that were also found in all of the other input arrays, duplicates are removed from the result
	
	Example:
	
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c");`
	`var a2 = new Array("c", "d", "e");`
	`gs.print(arrayUtil.intersect(a1, a2));`
	
	`*** Script: c`

Array	**union (`Array a, Array b, Array (n number of arrays can be supplied)`)**
	merge two or more arrays together
	
	Parameters:
	`Array a -`
	`Array b -`
	`Array (n number of arrays can be supplied) -`
	Returns:
	`Array` - returns an array of items from all of the input arrays, duplicates are removed from the result
	Example:
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c");`
	`var a2 = new Array("c", "d", "e");`
	`gs.print(arrayUtil.union(a1, a2));`
	
	`*** Script: a,b,c,d,e`

Array	**unique (`Array a1`)**
	remove duplicate items from an array
	
	Parameters:
	`Array a1 -`
	Returns:
	`Array -`
	Example:
	`var arrayUtil = new ArrayUtil();`
	`var a1 = new Array("a", "b", "c", "c", "b");`
	`gs.print(arrayUtil.unique(a1));`
	
	`*** Script: a,c,b`

DateTimeUtils

Overview

DateTimeUtils class is a collection of some additional date/time functions, provided via the Script Include **DateTimeUtils**.

For the core date/time functions, see GlideSystem Date and Time Functions

Where To Use

Methods from this script include are available to any server-side script. The DateTimeUtils class is also available through GlideAjax.

Method Summary

Method Summary	Description
int8ToGlideDateTime(int64)	Converts Microsoft AD integer8 DateTime format into GlideDateTime format. Integer8 is also known as Microsoft Filetime format. This method is commonly used when importing AD user's date fields, such as Expiration Date.
msToGlideDateTime()	Created to be used from client scripts to convert milliseconds to a GlideDateTime object,
formatCalendarDate()	Used by the calendar script from client, not intended to be called directly.

Method Detail

int8ToGlideDateTime(int64)

Converts Microsoft AD integer8 DateTime format into GlideDateTime format. Integer8 is also known as Microsoft Filetime format. This method is commonly used when importing AD user's date fields, such as Expiration Date.

Input Fields

Parameters:

- **int64** - a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

Output Fields

Returns: the GlideDateTime converted from the Integer8 date and time.

Example

```
//convert and set account expiration date from AD
//this is an example that could be used in an LDAP import transform map
 to import the LDAP account
//expires attribute to a customer created u_account_expires
GlideDateTime field
var dtUtil = new DateTimeUtils();
target.u_account_expires =
dtUtil.int8ToGlideDateTime(source.u_accountexpires);
```

msToGlideDateTime(int)

Created to be used from client scripts to convert milliseconds to a GlideDateTime object

Input Fields

Parameters:

- **int** - milliseconds.

Output Fields

Returns: the GlideDateTime object.

Example

```
//example script to call the method from a client
Replace MILLISECONDSVALUE with your variable
var ga = new GlideAjax('DateTimeUtils');
ga.addParam('sysparm_name','msToGlideDateTime');
ga.addParam('sysparm_value', MILLISECONDSVALUE);
ga.getXMLWait();
var newGDT = ga.getAnswer();
```

formatCalendarDate()

Used by the calendar script from client, not intended to be called directly.

Note: calendars are now legacy. Use Schedules instead.

FormInfoHeader

Overview

FormInfoHeader allows you to add an HTML message as a form Info Message

Where To Use

The addMessage function is available to server-side script. It is commonly used in record producers.

Method Summary	
Return Object	**Details**

void	**addMessage (`String message`)**
	Adds an HTML message to the form header, where form info messages are displayed
	Parameters:
	`String message` - message supports HTML tags
	Returns:
	`void -`
	Example:
	`gs.include("FormInfoHeader");`
	`var fi = new FormInfoHeader();`
	`fi.addMessage('This incident was opened on your behalf<br/>`
	`The IT department will contact you for further information or when the incident is resolved');`

J2js

Overview

J2JS script include allows you to convert java objects to javascript objects. If the given value is a Java object that can be converted to an equivalent JavaScript object, that conversion is performed and the result is returned. Otherwise the original Java object is returned.

Where To Use

The j2js class is available to server-side script.

Information

The specific conversions performed (in the order they are checked):

- Java Strings -> JavaScript strings
- Java Booleans -> JavaScript booleans
- Java Integers -> JavaScript numbers
- Java Longs -> JavaScript numbers
- Java Doubles -> JavaScript numbers
- Java Bytes -> JavaScript numbers
- Java Floats -> JavaScript numbers
- Java Shorts -> JavaScript numbers
- Java Characters -> JavaScript numbers
- Java arrays -> JavaScript Arrays with order preserved
- Java Lists -> JavaScript Arrays with order preserved
- Java Maps -> JavaScript Objects with the key/value pairs translated into property/value pairs
- Java Sets -> JavaScript Arrays in arbitrary order

Note that the conversions are performed recursively on the elements of arrays, lists, or collections. For example, given a Java ArrayList of ArrayLists of Strings, this will return a JavaScript Array of Arrays of strings.

Method Summary	
Return Object	**Details**

Object

j2js (`Object val`)

convert a java object from system code to a javascript object

Parameters:

`Object val` - A java object from system code such as a Packages call

Returns:

`Object` - JavaScript object if it can be converted, otherwise a Java object

Example:

```
var tu = new TableUtils("cmdb_ci_win_server");
var classes = tu.getHierarchy();
//getHierarchy returns a Java ArrayList, which is not exactly like a
JavaScript Array
//for example you cannot get length
gs.print("classes = " + classes);
gs.print("classes.length = " + classes.length);

//convert to a JavaScript Array
gs.include("j2js");
var jsClasses = j2js(classes);
gs.print("jsClasses = " + jsClasses);
gs.print("jsClasses.length = " + jsClasses.length);
```

```
*** Script: classes = [cmdb_ci_win_server, cmdb_ci_server, cmdb_ci_computer, cmdb_ci_hardware, cmdb_ci]
*** Script: classes.length = undefined
*** Script: jsClasses = cmdb_ci_win_server,cmdb_ci_server,cmdb_ci_computer,cmdb_ci_hardware,cmdb_ci
*** Script: jsClasses.length = 5
```

JSUtil

Overview

JSUtil is a class of shortcuts for common javascript routines.

Scoped applications are available starting with the Fuji release. Scripts created for scoped applications use the scoped API. In addition to the scoped API, ServiceNow script includes and business rules that are marked as Application = “global” and Accessible from = “All applications” can be used in scoped scripts.

The JSUtil class is not available in scoped applications.

Where To Use

The JSUtil class is available to server-side scripts.

Method Summary

Return Object	Details
boolean	**has(`Object item`)** **check if item is not null and is not undefined** **Parameters:** `Object item` - **Returns:** `boolean` - returns true if the given item is not null and is not undefined **Example:**

```
var x = "the quick brown fox";
var y = "";
var z;

gs.print("x = '" + x + "', JSUtil.has(x) = " + JSUtil.has(x));
gs.print("y = '" + y + "', JSUtil.has(y) = " + JSUtil.has(y));
gs.print("z = '" + z + "', JSUtil.has(z) = " + JSUtil.has(z));
```

```
*** Script: x = 'the quick brown fox', JSUtil.has(x) = true
*** Script: y = '', JSUtil.has(y) = true
*** Script: z = 'undefined', JSUtil.has(z) = false
```

boolean	**doesNotHave (`Object item`)**
	
	Returns true if the given item is null or is undefined (the logical inverse of .has(), above).
	
	Parameters:
	`Object item` -
	
	Returns:
	`boolean` - Returns true if the given item is null or is undefined (the logical inverse of .has(), above).
	
	Example:
	`var x = "the quick brown fox";`
	`var y = "";`
	`var z;`
	
	`gs.print("x = '" + x + "', JSUtil.doesNotHave(x) = " + JSUtil.doesNotHave(x));`
	`gs.print("y = '" + y + "', JSUtil.doesNotHave(y) = " + JSUtil.doesNotHave(y));`
	`gs.print("z = '" + z + "', JSUtil.doesNotHave(z) = " + JSUtil.doesNotHave(z));`
	
	`*** Script: x = 'the quick brown fox', JSUtil.doesNotHave(x) = false`
	`*** Script: y = '', JSUtil.doesNotHave(y) = false`
	`*** Script: z = 'undefined', JSUtil.doesNotHave(z) = true`

boolean	**nil (`Object item`)**
	
	Returns true if the given item is null, undefined, or evaluates to the empty string.
	
	Parameters:
	`Object item` -
	
	Returns:
	`boolean` - Returns true if the given item is null, undefined, or evaluates to the empty string.
	
	Example:
	`var x = "the quick brown fox";`
	`var y = "";`
	`var z;`
	
	`gs.print("x = '" + x + "', JSUtil.nil(x) = " + JSUtil.nil(x));`
	`gs.print("y = '" + y + "', JSUtil.nil(y) = " + JSUtil.nil(y));`
	`gs.print("z = '" + z + "', JSUtil.nil(z) = " + JSUtil.nil(z));`
	
	`*** Script: x = 'the quick brown fox', JSUtil.nil(x) = false`
	`*** Script: y = '', JSUtil.nil(y) = true`
	`*** Script: z = 'undefined', JSUtil.nil(z) = true`

boolean	**notNil (`Object item`)** **Returns true if the given item exists and is not empty (the logical inverse of .nil(), above).** **Parameters:** `Object item` - **Returns:** `boolean` - Returns true if the given item exists and is not empty (the logical inverse of .nil(), above). **Example:**

```
var x = "the quick brown fox";
var y = "";
var z;

gs.print("x = '" + x + "', JSUtil.notNil(x) = " + JSUtil.notNil(x));
gs.print("y = '" + y + "', JSUtil.notNil(y) = " + JSUtil.notNil(y));
gs.print("z = '" + z + "', JSUtil.notNil(z) = " + JSUtil.notNil(z));
```

```
*** Script: x = 'the quick brown fox', JSUtil.notNil(x) = true
*** Script: y = '', JSUtil.notNil(y) = false
*** Script: z = 'undefined', JSUtil.notNil(z) = false
```

boolean	**instance_of (`Object item, String klass`)** **Returns true if the given item is a member of the given class.** **For JavaScript objects, this method behaves exactly like the JavaScript operator "instanceof", but it also supports Java objects** **Parameters:** `Object item` - object to be tested `String klass` - the class to be tested (for Java objects, must be the complete class name, like "java.util.ArrayList") **Returns:** `boolean` - Returns true if the given item is a member of the given class. **Example:**

```
var a = ["a","b","c"];
var b = 10;
var c = new GlideRecord("incident");

gs.print("JSUtil.instance_of(a, 'Array') = " + JSUtil.instance_of(a,
'Array'));
gs.print("JSUtil.instance_of(a, 'String') = " + JSUtil.instance_of(a,
'String'));

gs.print("JSUtil.instance_of(b, 'String') = " + JSUtil.instance_of(b,
'String'));

gs.print("JSUtil.instance_of(c, 'GlideRecord') = " +
JSUtil.instance_of(c, 'GlideRecord'));
```

```
*** Script: JSUtil.instance_of(a, 'Array') = true
*** Script: JSUtil.instance_of(a, 'String') = false
*** Script: JSUtil.instance_of(b, 'String') = false
*** Script: JSUtil.instance_of(c, 'GlideRecord') = true
```

String

type_of (`Object value`)

Returns the type of the given value as a string

Parameters:

`Object value` - object you wish to get the type on

Returns:

`String` - Returns the type of the given value as a string, as follows:

- 'null' if the given value is null or undefined
- 'string' if the given value is a primitive string or a String wrapper instance
- 'number' if the given value is a primitive number or a Number wrapper instance
- 'boolean' if the given value is a primitive boolean or a Boolean wrapper instance
- 'function' if the given value is a function
- 'object' otherwise

Example:

```
var a = ["a","b","c"];
var b = 10;
var c = new GlideRecord("incident");
var d = true;
var e;

gs.print("JSUtil.type_of(a) = " + JSUtil.type_of(a));
gs.print("JSUtil.type_of(b) = " + JSUtil.type_of(b));
gs.print("JSUtil.type_of(c) = " + JSUtil.type_of(c));
gs.print("JSUtil.type_of(d) = " + JSUtil.type_of(d));
gs.print("JSUtil.type_of(e) = " + JSUtil.type_of(e));
```

```
*** Script: JSUtil.type_of(a) = object
*** Script: JSUtil.type_of(b) = number
*** Script: JSUtil.type_of(c) = object
*** Script: JSUtil.type_of(d) = boolean
*** Script: JSUtil.type_of(e) = null
```

boolean	**isJavaObject (`value`)** **Returns true if the given value is an instance of a Java class** **Parameters:** `value` - object to test **Returns:** `boolean` - Returns true if the given value is an instance of a Java class **Example:**

```
var tu = new TableUtils("incident");
var classes = tu.getHierarchy(); //Java ArrayList
var tables = ["task, incident"]; //JavaScript Array

gs.print("JSUtil.isJavaObject(classes) = " +
JSUtil.isJavaObject(classes));
gs.print("JSUtil.isJavaObject(tables) = " +
JSUtil.isJavaObject(tables));
```

```
*** Script: JSUtil.isJavaObject(classes) = true
*** Script: JSUtil.isJavaObject(tables) = false
```

boolean

toBoolean (`item`)

Coerces the given item to a boolean

Parameters:

`item` - value to convert to a boolean

Returns:

`boolean` - Coerces the given item to a boolean. If the given item is a boolean, it is passed through. Non-zero numbers return true. Null or undefined returns false. Strings return true only if exactly equal to 'true'.

Example:

```
var zero = 0;
var one = 1;
var number = 12;
var trueBoolean = true;
var trueString = "true";
var otherString = "random text";

gs.print("JSUtil.toBoolean(zero) = " + JSUtil.toBoolean(zero));
gs.print("JSUtil.toBoolean(one) = " + JSUtil.toBoolean(one));
gs.print("JSUtil.toBoolean(number) = " + JSUtil.toBoolean(number));
gs.print("JSUtil.toBoolean(trueBoolean) = " +
JSUtil.toBoolean(trueBoolean));
gs.print("JSUtil.toBoolean(trueString) = " +
JSUtil.toBoolean(trueString));
gs.print("JSUtil.toBoolean(otherString) = " +
JSUtil.toBoolean(otherString));
```

```
*** Script: JSUtil.toBoolean(zero) = false
*** Script: JSUtil.toBoolean(one) = true
*** Script: JSUtil.toBoolean(number) = true
*** Script: JSUtil.toBoolean(trueBoolean) = true
*** Script: JSUtil.toBoolean(trueString) = true
*** Script: JSUtil.toBoolean(otherString) = false
```

boolean	**getBooleanValue (`GlideRecord gr, String field`)**
	
	Returns the value in a boolean GlideRecord field
	
	Parameters:
	`GlideRecord gr` -
	`String field` - field to get boolean value from
	Returns:
	`boolean` - Returns the value in a boolean GlideRecord field, returns true if value of field is true, "true", 1, or "1"
	Example:
	`var inc = new GlideRecord("incident");`
	`//get an active incindent`
	`inc.addActiveQuery();`
	`inc.setLimit(1);`
	`inc.query();`
	`inc.next();`
	
	`gs.print(JSUtil.getBooleanValue(inc, "active"));`
	
	`*** Script: true`

void

logObject (`Object obj, [String name]`)

Logs all the properties in the given object: name, type, and value. Useful when troubleshooting objects.

Parameters:

`Object obj` - Object to enumerate properties on

`[String name]` - Optional name for the logged object

Returns:

`void` - Output is written to console if you are running from a background script or have debug logging enables. They are also written to the system log.

Example:

```
var arr = ["a","b","c"];

var inc = new GlideRecord("incident");
//get an active incindent
inc.addActiveQuery();
inc.setLimit(1);
inc.query();
inc.next();

JSUtil.logObject(arr, "arr");
JSUtil.logObject(inc, "inc");
```

```
*** Script: Log Object: arr
  Array of 3 elements
    [0]: string = a
    [1]: string = b
    [2]: string = c
*** Script: Log Object: inc
  GlideRecord('incident') @ INC0000002
```

String

escapeText (`String text`)

escape invalid xml characters such as < > &

Parameters:

`String text` -

Returns:

`String` - escaped string

Example:

```
var html = "<b>This is my title</b>";

gs.print(JSUtil.escapeText(html));
```

```
*** Script: <b>This is my title</b>
this is the actual value of the return, if it is being displayed in the
 application the page will render the brackets back so it will appear
that is was not escaped, but it really was.
```

String	**unescapeText (`String text`)**
	
	restore characters that have been escaped
	
	Parameters:
	`String text -`
	Returns:
	`String` - unescaped string
	Example:
	`var html = "&lt;b&gt;This is my title&lt;/b&gt;";`
	`gs.print(JSUtil.unescapeText(html));`
	
	`*** Script: <b>This is my title</b>`
	`this is the actual value of the return, if it is being displayed in the application the page will render the html tags and display the text in bold.`

String	**escapeAttr (`String text`)**
	
	escape ampersands commonly used to define URL attributes
	
	Parameters:
	`String text -`
	Returns:
	`String` - string with ampersands properly escaped
	Example:
	`var attr = "sysparm_query=active=true&sysparm_view=special";`
	`gs.print(JSUtil.escapeAttr(attr));`
	
	`sysparm_query=active=true&sysparm_view=special`
	`this is the actual value of the return, if it is being displayed in the application the page will render the escaped ampersand with a single ampersand`

String	**unescapeAttr (`String text`)**
	restore ampersands from escaped text
	Parameters:
	`String text` -
	Returns:
	`String` - unescaped string
	Example:
	`var attr = "sysparm_query=active=true&amp;sysparm_view=special";`
	`gs.print(JSUtil.unescapeAttr(attr));`
	`*** Script: sysparm_query=active=true&sysparm_view=special`

RecordToHTML

Overview

RecordToHTML is a utility class to turn a record in a table into HTML.

For example, the following script:

```
var r2html = new
RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5",
"incident: ${number}-${short_description}", true);
gs.print(r2html.toString());
```

Produces the following output:

```
incident: INC00005-CPU load high for over 10 minutes
```

Where To Use

The RecordToHTML class is available to server-side scripts.

Method Summary	
Return Object	**Details**

void

initialize (`String table, String sys_id, String pattern, boolean link`)

called when instantiating a new object of this class

Parameters:

`String table` - name of the table from which this record comes

`String sys_id` - the sys_id of the record

`String pattern` - Pattern of the string we want to generate. The pattern may include ${} escapes for fields whose values should be included. For instance, the pattern "sys_id: ${sys_id}" would substitute the actual sys_id for the escape.

`boolean link` - {{{parameter4desc}}}

Returns:

`void` -

Example:

```
var r2html = new
RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5", "incident:
${number}-${short_description}", true);
gs.print(r2html.toString());
```

Output:

```
incident:  INC00005-CPU load high for over 10 minutes
```

void

setValue (`String name, String value`)

Set a field value. This is useful if the calling code has already read the underlying GlideRecord, and wants to avoid the overhead of another database read.

Parameters:

`String name` - name of a field

`String value` - value to set the field to

Returns:

`void` -

Example:

```
var r2html = new
RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5", "incident:
${number}-${short_description} (${user})", true);
r2html.setValue("user",gs.getUserName());
gs.print(r2html.toString());
```

```
incident:  INC00005-CPU load high for over 10 minutes (john.roberts)
```

String	**toString ()**
	
	converts this instance to a string
	
	Returns:
	
	`String` - HTML output for the record
	
	Example:
	
	`var r2html = new RecordToHTML("incident","e8e875b0c0a80164009dc852b4d677d5", "incident:${number}-${short_description}", true);`
	`gs.print(r2html.toString());`
	
	`incident:  INC00005-CPU load high for over 10 minutes`

TableUtils

Note: *This article applies to Fuji and earlier releases. For more current information, see Server API Reference* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.'**

Overview

TableUtils is a class of shortcuts for accessing table related information

Where To Use

The TableUtils class is available to server-side scripts.

Method Summary

Return Object

void

boolean

void

void

void

ArrayList

ArrayList

ArrayList

String

ArrayList

boolean

boolean

boolean

Details

initialize (`String tableName`)

called when instantiating the TableUtils class

Parameters:

`String tableName` - valid name of a table

Returns:

`void` -

Example:

```
var tu = new TableUtils("incident");
```

tableExists ()

checks to see if a table exists

Returns:

`boolean` - true if the table exists, otherwise false

Example:

```
var table = new TableUtils("my_table");
gs.print("Does 'my_table' exist? " + table.tableExists());
```

```
*** Script: Does 'my_table' exist? false
```

drop (`String tableName`)

drops a database table (also see dropAndClean)

Parameters:

`String tableName` - name of the table to drop

Returns:

`void` -

Example:

Warning: Use with extreme caution, dropping a database table will permanently delete the table and all of the data. Also see dropAndClean. If the table is extended use dropTableAndExtensions.

```
var tu = new TableUtils();
tu.drop("table_that_will_be_lost_forever");
```

```
*** Script: dropping table table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table table_that_will_be_lost_forever
```

dropAndClean (`String tableName`)

drops a database table and cleans up references to the table

Parameters:

`String tableName` - name of the table to drop

Returns:

`void -`

Example:

Warning: Use with extreme caution, dropping a database table will permanently delete the table and all of the data. If the table is extended use dropTableAndExtensions.

```
var tu = new TableUtils();
tu.dropAndClean("table_that_will_be_lost_forever");
```

```
*** Script: dropping table table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table table_that_will_be_lost_forever
*** Script: removing gauges for table_that_will_be_lost_forever
*** Script: removing forms for table_that_will_be_lost_forever
*** Script: removing styles for table_that_will_be_lost_forever
*** Script: removing forms sections for table_that_will_be_lost_forever
*** Script: removing lists for table_that_will_be_lost_forever
*** Script: removing related lists for table_that_will_be_lost_forever
*** Script: removing references to table_that_will_be_lost_forever
*** Script: removing dictionary entries for table_that_will_be_lost_forever
Background message, type:info, message: Table deleted
```

dropTableAndExtensions (`String tableName`)

drops a database table, all of it's extended tables and cleans up references to the tables

Parameters:

`String tableName` - name of the parent table to drop

Returns:

`void -`

Example:

Warning: Use with extreme caution, dropping a database table will permanently delete the table and all of the data.

```
var tu = new TableUtils();
tu.dropAndClean("table_that_will_be_lost_forever");
```

```
*** Script: dropping table parent_table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table ext_table_that_will_be_lost_forever
*** Script: removing gauges for ext_table_that_will_be_lost_forever
*** Script: removing forms for ext_table_that_will_be_lost_forever
*** Script: removing styles for ext_table_that_will_be_lost_forever
*** Script: removing forms sections for ext_table_that_will_be_lost_forever
*** Script: removing lists for ext_table_that_will_be_lost_forever
*** Script: removing related lists for ext_table_that_will_be_lost_forever
*** Script: removing references to ext_table_that_will_be_lost_forever
*** Script: removing dictionary entries for ext_table_that_will_be_lost_forever
Background message, type:info, message: Table deleted
*** Script: dropping table parent_table_that_will_be_lost_forever
Starting cache flush
Cache flush complete
TABLE DROP: admin dropped table parent_table_that_will_be_lost_forever
*** Script: removing gauges for parent_table_that_will_be_lost_forever
*** Script: removing forms for parent_table_that_will_be_lost_forever
*** Script: removing styles for parent_table_that_will_be_lost_forever
*** Script: removing forms sections for parent_table_that_will_be_lost_forever
*** Script: removing lists for parent_table_that_will_be_lost_forever
*** Script: removing related lists for parent_table_that_will_be_lost_forever
*** Script: removing references to parent_table_that_will_be_lost_forever
*** Script: removing dictionary entries for parent_table_that_will_be_lost_forever
Background message, type:info, message: Table deleted
```

getTables ()

get the table hierarchy

Returns:

`ArrayList` - Java ArrayList of table names in the parent hierarchy

Example:

```
var table = new TableUtils("incident");
gs.print(table.getTables());
```

```
*** Script: [incident, task]
```

getTableExtensions ()

get the list of tables that extend a table

Returns:

`ArrayList` - Java ArrayList of table names that extend a table

Example:

```
var table = new TableUtils("task");
gs.print(table.getTableExtensions());
```

```
*** Script: [incident, issue, kb_submission, sysapproval_group, change_request, change_request_imac, sc_task,
problem, sc_req_item, ticket, ast_transfer_order, planned_task, change_task, change_phase, sc_request]
```

getAllExtensions ()

get the list of tables that extend a table, includes the base table

Returns:

`ArrayList` - Java ArrayList of table names that extend a table, includes the base table

Example:

```
var table = new TableUtils("task");
gs.print(table.getAllExtensions());
```

```
*** Script: [task, incident, issue, kb_submission, sysapproval_group, change_request, change_request_imac, sc_task,
problem, sc_req_item, ticket, ast_transfer_order, planned_task, change_task, change_phase, sc_request]
```

getAbsoluteBase ()

get the base table name that a table was extended from

Returns:

`String` - base table name

Example:

```
var table = new TableUtils("cmdb_ci_server");
gs.print(table.getAbsoluteBase());
```

```
*** Script: cmdb_ci
```

getHierarchy ()

returns a list of all classes participating in the hierarchy of the given table.

Returns:

`ArrayList` - returns a list of all classes participating in the hierarchy of the given table.

Example:

```
var table = new TableUtils("cmdb_ci_server");
gs.print(table.getHierarchy());
```

```
*** Script: [cmdb_ci_server, cmdb_ci_computer, cmdb_ci_hardware, cmdb_ci, cmdb_ci_mainframe, cmdb_ci_linux_server,
cmdb_ci_mainframe_lpar, cmdb_ci_esx_server, cmdb_ci_unix_server, cmdb_ci_solaris_server, cmdb_ci_hpux_server,
cmdb_ci_aix_server, cmdb_ci_osx_server, cmdb_ci_netware_server, cmdb_ci_win_server]
```

hasExtensions ()

check to see if a table has extension tables

Returns:

`boolean` - returns true if the table has extensions e.g task has extensions but incident does not

Example:

```
var table = new TableUtils("cmdb_ci_server");
gs.print(table.hasExtensions());
```

```
*** Script: true
```

isBaseClass ()

check to see if a table has no parents and has extensions.
Task is a base class (not extended from another table, and it has tables extended from it), but sys_user is not (since it has no parents, but does not have exentions)

Returns:

`boolean` -

Example:

```
var table = new TableUtils("task");
gs.print("Task is base class: " + table.isBaseClass());

var table = new TableUtils("sys_user");
gs.print("User is base class: " + table.isBaseClass());
```

```
*** Script: Task is base class: true
*** Script: User is base class: false
```

isSoloClass ()

check to see if table has no parents and no extensions

Returns:

`boolean` - true if table has no parent (not extended from another table) and no tables are extended from it

Example:

```
var table = new TableUtils("task");
gs.print("task is solo class: " + table.isSoloClass());

var table = new TableUtils("cmdb_ci_win_server");
gs.print("cmdb_ci_win_server is solo class: " + table.isSoloClass());

var table = new TableUtils("sys_user");
gs.print("sys_user is solo class: " + table.isSoloClass());
```

```
*** Script: task is solo class: false
*** Script: cmdb_ci_win_server is solo class: false
*** Script: sys_user is solo class: true
```

App-Specific

Cost Management

ExpenseLine

Overview

The ExpenseLine API is included with the Cost Management Plugin as a script include record. It is used by various cost management processes and can also be used for generating expense line (fm_expense_line) records from your own server-side scripts.

Methods

Method Summary	
Return Value	**Details**
ExpenseLine	**initialize**(`GlideRecord source, Decimal amount,` *`String description, GlideRecord costSource`*) Instantiates new ExpenseLine object
	setCostSource(`GlideRecord costSource`) Identifies the source rate card or distribution cost that was the source of expense line generation. This is not the source (CI, task) of the expense.
	setDescription(`String description`*)* Defines the a description of the expense
	setRecurring(`boolean recurring`) Flags the expense as recurring by setting the `recurring` field to true
	setParent(`GlideRecord expense`) Sets the parent field on the expense line. This is generally only used by the system when generating indirect expenses such as business service aggregated expenses
	setSummaryType(`String summaryType`) Sets a value for the expense line `summary_type` field
boolean	**createExpense**() Creates the expense line record based on the parameters provided
	processCIParents() Used internally by the createExpense method to process CI relationships when the expense source is a cmdb_ci type record.

initialize

initialize(`GlideRecord source, Decimal amount,` *`String description, GlideRecord costSource`*)

Instantiates a new ExpenseLine object with basic required expense line data.

Parameters:

`GlideRecord source` - Required GlideRecord identifying the source of the expense

`Decimal amount` - Required Decimal identifying the amount of the expense

`String description` - *Optional description of the expense*

`GlideRecord costSource` - *Optional GlideRecord identifying the source record that generated the expense line. This is generally only used for system generated expense lines (CI rate card, distribution costs, and task rate card processing)*

Returns:

`ExpenseLine` object

---

setCostSource

setCostSource(`GlideRecord costSource`)

Identifies the source rate card or distribution cost that was the source of expense line generation. This is not the source (CI, task) of the expense.

Parameters:

`costSource` - GlideRecord of CI rate card cost, distribution cost, or task rate card. This is generally only used for system generated expense lines.

---

setDescription

setDescription(`String description`)

Defines the a description of the expense.

Parameters:

`description` - Description of expense.

---

setRecurring

setRecurring(`boolean recurring`)

Flags the expense as recurring by setting the `recurring` field to true. Expense lines are set to false by default so there is no need to call setRecurring(false).

Parameters:

`recurring` - true to identify expense line as a recurring expense

setParent

setParent(`GlideRecord expense`)

Sets the `parent` field on the expense line. This is generally only used by the system when generating indirect expenses such as business service aggregated expenses.

Parameters:

`expense` - parent expense line record

setSummaryType

setSummaryType(`String summaryType`)

Sets a value for the expense line `summary_type` field.

Parameters:

`summaryType` - Typically you would set it to a value already specified in the expense line summary type field choice list

createExpense

createExpense()

Creates the expense line record based on the parameters provided.

Returns:

`boolean` true if the expense line was successfully created

processCIParents

processCIParents()

Used internally by the `createExpense` method to process CI relationships when the expense source is a cmdb_ci type record.

Example

```
//get some random CI to be used as an expense source
var ci = new GlideRecord("cmdb_ci_server");
ci.query();
ci.next();

//create expense line
var exp = new ExpenseLine(ci, 234.56, "Test expense line");
exp.setSummaryType("run_business");
var success = exp.createExpense();
gs.print("Successful? " + success);
```

ExpenseAllocation

Overview

The ExpenseAllocation API is included with the Cost Management Plugin as a script include record. It is used by various cost management processes and can also be used for generating custom expense allocation records (fm_expense_allocation) from scripted expense allocation rules.

Methods

Method Summary	
Return Value	**Details**
ExpenseAllocation	**initialize**(`GlideRecord expense, GlideRecord rule`) Instantiates new ExpenseAllocation object, this is not needed if scripting advanced allocation rules. This object is already available as **allocation** variable.
boolean	**createAllocation**(`GlideRecord target, Decimal amount`) Creates an expense allocation record

initialize

initialize(`GlideRecord expense, GlideRecord rule`)

Instantiates a new ExpenseAllocation object with basic required allocation data. Called when using *new ExpenseAllocation()*.

Parameters:

`GlideRecord source` - Required GlideRecord identifying the source of the expense

`Decimal amount` - Required Decimal identifying the amount of the expense

`String description` - *Optional description of the expense*

`GlideRecord costSource` - *Optional GlideRecord identifying the source record that generated the expense line. This is generally only used for system generated expense lines (CI rate card, distribution costs, and task rate card processing)*

Returns:

`ExpenseLine` object

---

createAllocation

createAllocation(`GlideRecord target, Decimal amount`)

Creates an expense allocation (fm_expense_allocation) record referencing the parameters provided during instantiation and this method.

Parameters:

`GlideRecord target` - GlideRecord target of the allocation, for example a cost center record to allocate an expense to

`Decimal amount` - the amount of the allocation

Returns:

`boolean` true if the expense line was successfully created

Example

```
var allocation = new ExpenseAllocation(expenseGlideRecord,
ruleGlideRecord);
allocation.createAllocation(costCenterGlideRecord, 2345.67);
```

Change Collision

Change Collision

Overview

The Change Management Collision Detector Plugin includes three Script Includes, each of which contain helper functions specific to the plugin, when scripting on the server side or using AJAX calls on the client.

The three script includes are:

- ChangeCollisionHelper
- ChangeConflict
- ChangeConflictHandler

ChangeCollisionHelper

Method Summary	
Return Object	**Details**
boolean	**isDateInCiMaintenanceWindows (`startDate, endDate, maintenanceWindow`)** **Checks if the time span defined by *startDate* and endDate *falls in the CI's maintenance window*** **Parameters:** `startDate` - The beginning date of a time span. `endDate` - The ending date of a time span. `maintenanceWindow` - A Configuration Item's sys_id. **Returns:** `boolean` - An array of CIs.
	getCiMaintenanceSchedule (`ci`) **Gets the Maintenance Schedule for a CI.** **Parameters:** `ci` - A Configuration Item's sys_id.

array	**getBlackoutsByDate (`startDate, enDate`)** **Gets any blackout that overlap the period defined by *startDate* and *endDate*.** **Parameters:** `startDate` - The beginning date of a time span. `enDate` - The ending date of a time span. **Returns:** `array` - An array of blackouts (blackoutId:stringSpan).
array changeIds	**getChangesWithAffectedCi (`ci, startDate, enDate`)** **Get changes scheduled in the timespan (defined by *startDate* and *endDate*) that have the given CI in their *Affected CIs* list.** **Parameters:** `ci` - A Configuration Item's sys_id. `startDate` - The beginning date of a time span. `enDate` - The ending date of a time span. **Returns:** `array changeIds` - An array of sys_ids for Change records.
array changeIds	**getChangesWithCi (`ci, startDate, enDate`)** **Get the changes that are in the timespan (*startDate*, *endDate*) and that are linked to the given CI.** **Parameters:** `ci` - A Configuration Item's sys_id. `startDate` - The beginning date of a time span. `enDate` - The ending date of a time span. **Returns:** `array changeIds` - An array of sys_ids for Change records.
array	**getAffectedCisByChangeId (`changeId`)** **Gets the *Affected CI* sys_ids for the given change.** **Parameters:** `changeId` - A Change Record's sys_id **Returns:** `array` - An array of sys_ids of Affected CIs.
	addCiToChangeAffectedCis (`ci, changeid`) **Add CI to the change's *Affected CI* list.** **Parameters:** `ci` - A Configuration Item's sys_id. `changeid` - A Change Record's sys_id.

boolean	**isCiInAffectedCis (`ci, changeid`)** **Check if an CI is already in the change's *Affected CIs* list.** **Parameters:** `ci` - A Configuration Item's sys_id. `changeid` - A Change Record's sys_id. **Returns:** `boolean` - True if the CI already in the change's *Affected CIs* list, false if not.
array	**getDependants (`ci`)** **Get all the CIs that depend on the given CI.** **Parameters:** `ci` - A Configuration Item's sys_id. **Returns:** `array` - An array of CIs.
array	**getDependencies (`ci`)** **Get all the CIs that the given CI depends on.** **Parameters:** `ci` - A Configuration Item's sys_id. **Returns:** `array` - An array of CIs.

ChangeConflict

Method Summary	
Return Object	**Details**
	initialize (`configurationItemId, changeId, type`) **Initializes a ChangeConflict object.** **Parameters:** `configurationItemId` - A Configuration Item's sys_id. `changeId` - A Change Request's sys_id. `type` - A value from the conflict's Type choice list. **Example:** `var cc = new ChangeConflict(82992eb60ad337024fbb6d06a866c636,8799385b0a0a2c3e14c041a7f5414266,not_in_maintenance_window);`

string	**toString ()** **Returns a String representation of the conflict** **Returns:** `string` - The string representation of the conflict.

ChangeConflictHandler

Method Summary	
Return Object	**Details**
	initialize () **Initializes a Change Conflict Container array.**
	addChangeConflict (`changeConflict`) **Adds the Change Conflict to a Change Conflict Container.** **Parameters:** `changeConflict` - The sys_id of a Change Conflict.
changeConflictContainer	**getConflicts ()** **Gets an array of Change Conflicts from a Change Conflict Container.** **Returns:** `changeConflictContainer` - An array of Change Conflicts.
	saveConflicts () **Writes out the Change Conflicts in a Change Conflict Container array to individual Change Conflict records.**
	deleteConflictsByChangeId (`changeID`) **Deletes conflicts that are associated with the same Change Request (by sys_id).** **Parameters:** `changeID` - A sys_id of a Change Request.

Article Sources and Contributors

GlideRecord *Source*: http://wiki.servicenow.com/index.php?oldid=250207 *Contributors*: Anat.kerry, Andrew.Kincaid, Brent.bahry, Carleen.greenlee, Chris.henson, Chuck.tomasi, David.Bailey, Eric.jacobson, Fuji.publishing.user, G.yedwab, George.rawlins, Guy.yedwab, Jeremy.norris, Jim.uebbing, John.ramos, John.roberts, Joseph.messerschmidt, Kevin.pickard, Mark.stanger, Neola, Phillip.salzman, Rachel.sienko, Russ.sarbora, Steven.wood, Tom.dilatush, Vaughn.romero, Wallymarx

GlideSystem *Source*: http://wiki.servicenow.com/index.php?oldid=250606 *Contributors*: Amy.bowman, Anat.kerry, CapaJC, Chuck.tomasi, David.Bailey, Fred.luddy, Fuji.publishing.user, G.yedwab, George.rawlins, Guy.yedwab, Jim.uebbing, John.ramos, Joseph.messerschmidt, Mark.stanger, Neola, Phillip.salzman, Rachel.sienko, Steven.wood

GlideElement *Source*: http://wiki.servicenow.com/index.php?oldid=250298 *Contributors*: Amy.bowman, Andrew.Kincaid, CapaJC, Emily.partridge, Eric.jacobson, Fred.luddy, Fuji.publishing.user, G.yedwab, George.rawlins, Gflewis, Guy.yedwab, Jim.uebbing, John.ramos, Joseph.messerschmidt, Neola, Rachel.sienko, Steven.wood

GlideAggregate *Source*: http://wiki.servicenow.com/index.php?oldid=250293 *Contributors*: Anat.kerry, Don.Goodliffe, G.yedwab, George.rawlins, Guy.yedwab, Jim.uebbing, John.ramos, Joseph.messerschmidt, Mark.stanger, Matt.kilbride, Neola, Rachel.sienko, Steven.wood, Vaughn.romero, Vhearne

GlideForm (g form) *Source*: http://wiki.servicenow.com/index.php?oldid=250605 *Contributors*: Amy.bowman, Anat.kerry, Ashley.robinson, Bsweetser, David.Bailey, Emily.partridge, Fuji.publishing.user, G.yedwab, Gadi.yedwab, George.rawlins, Guy.yedwab, Jim.uebbing, John.ramos, Joseph.messerschmidt, Maneesha.Nanda, Mark.stanger, Michael.randall, Neola, Pat.Casey, Phillip.salzman, Rachel.sienko, Roy.lagemann, Steven.wood, Suzanne.smith, Wallymarx

GlideUser (g user) *Source*: http://wiki.servicenow.com/index.php?oldid=250608 *Contributors*: Emily.partridge, George.rawlins, Guy.yedwab, Heidi.schnakenberg, John.ramos, Joseph.messerschmidt, Liz.malone, Neola

CIUtils *Source*: http://wiki.servicenow.com/index.php?oldid=130167 *Contributors*: Emily.partridge, G.yedwab, George.rawlins, John.roberts, Neola

ArrayUtil *Source*: http://wiki.servicenow.com/index.php?oldid=130164 *Contributors*: Emily.partridge, G.yedwab, John.roberts, Joseph.messerschmidt, Neola, Rachel.sienko

DateTimeUtils *Source*: http://wiki.servicenow.com/index.php?oldid=227658 *Contributors*: George.rawlins, Guy.yedwab, John.roberts, Joseph.messerschmidt, Neola

FormInfoHeader *Source*: http://wiki.servicenow.com/index.php?oldid=151756 *Contributors*: George.rawlins, Guy.yedwab, John.roberts, Joseph.messerschmidt, Mark.stanger, Neola

J2js *Source*: http://wiki.servicenow.com/index.php?oldid=151754 *Contributors*: George.rawlins, Guy.yedwab, John.roberts, Joseph.messerschmidt, Neola

JSUtil *Source*: http://wiki.servicenow.com/index.php?oldid=249455 *Contributors*: Emily.partridge, George.rawlins, Guy.yedwab, Jim.uebbing, John.roberts, Mark.stanger, Neola, Peter.smith

RecordToHTML *Source*: http://wiki.servicenow.com/index.php?oldid=169066 *Contributors*: David.Bailey, Emily.partridge, George.rawlins, Guy.yedwab, John.roberts, Joseph.messerschmidt, Neola

TableUtils *Source*: http://wiki.servicenow.com/index.php?oldid=250942 *Contributors*: Emily.partridge, George.rawlins, Guy.yedwab, John.ramos, John.roberts, Neola, Rachel.sienko, Steven.wood

ExpenseLine *Source*: http://wiki.servicenow.com/index.php?oldid=104255 *Contributors*: G.yedwab, Guy.yedwab, Joe.Westrich, John.roberts, Joseph.messerschmidt, Neola, Rachel.sienko

ExpenseAllocation *Source*: http://wiki.servicenow.com/index.php?oldid=104233 *Contributors*: G.yedwab, Guy.yedwab, Joe.Westrich, John.roberts, Joseph.messerschmidt, Neola, Rachel.sienko

Change Collision *Source*: http://wiki.servicenow.com/index.php?oldid=248190 *Contributors*: David.Bailey, Davida.hughes, Emily.partridge, G.yedwab, Guy.yedwab, Joseph.messerschmidt, Julie.komrosky, Neola, Rachel.sienko

Image Sources, Licenses and Contributors

Image:Warning.gif *Source*: http://wiki.servicenow.com/index.php?title=File:Warning.gif *License*: unknown *Contributors*: CapaJC

Image:addDecoration_method.png *Source*: http://wiki.servicenow.com/index.php?title=File:AddDecoration_method.png *License*: unknown *Contributors*: George.rawlins

Image:SetDisplayExampleFieldDisplayed.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:SetDisplayExampleFieldDisplayed.PNG *License*: unknown *Contributors*: Neola

Image:SetDisplayExampleFieldHidden.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:SetDisplayExampleFieldHidden.PNG *License*: unknown *Contributors*: Neola

Image:SetVisibleExampleFieldDisplayed.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:SetVisibleExampleFieldDisplayed.PNG *License*: unknown *Contributors*: Neola

Image:SetVisibleExampleFieldHidden.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:SetVisibleExampleFieldHidden.PNG *License*: unknown *Contributors*: Neola

Image:ShowFieldMsgInfo.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:ShowFieldMsgInfo.PNG *License*: unknown *Contributors*: Neola

Image:ShowFieldMsgError.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:ShowFieldMsgError.PNG *License*: unknown *Contributors*: Neola

Image:AddErrorMessage.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:AddErrorMessage.PNG *License*: unknown *Contributors*: Neola

Image:AddInfoMessage.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:AddInfoMessage.PNG *License*: unknown *Contributors*: Neola

Image:UserNameFluddy.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:UserNameFluddy.PNG *License*: unknown *Contributors*: Neola

Image:UserID.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:UserID.PNG *License*: unknown *Contributors*: Neola

Image:FirstName.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:FirstName.PNG *License*: unknown *Contributors*: Neola

Image:LastName.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:LastName.PNG *License*: unknown *Contributors*: Neola

Image:LoginLanguageButton.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:LoginLanguageButton.PNG *License*: unknown *Contributors*: Neola

Image:LoginLanguageAlert.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:LoginLanguageAlert.PNG *License*: unknown *Contributors*: Neola

Image:GetFullName.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:GetFullName.PNG *License*: unknown *Contributors*: Neola

Image:HasRoleAdmin.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:HasRoleAdmin.PNG *License*: unknown *Contributors*: Neola

Image:HasRoleExactly.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:HasRoleExactly.PNG *License*: unknown *Contributors*: Neola

Image:HasRoleFromList.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:HasRoleFromList.PNG *License*: unknown *Contributors*: Neola

Image:HasRoles.PNG *Source*: http://wiki.servicenow.com/index.php?title=File:HasRoles.PNG *License*: unknown *Contributors*: Neola

Image:Caution-diamond.png *Source*: http://wiki.servicenow.com/index.php?title=File:Caution-diamond.png *License*: unknown *Contributors*: John.roberts, Publishing.user