<https://developer.servicenow.com/dev.do#!/reference/api/vancouver/server/no-namespace/c_GlideRecordScopedAPI#r_ScopedGlideRecordIsNewRecord?navFilter=isne>

<https://developer.servicenow.com/dev.do#!/reference/api/vancouver/server_legacy/c_AbstractDBObject>

Tables:

|  |  |
| --- | --- |
| task\_sla | SLA Calculation |
| incident\_alert |  |
| incident\_task |  |
| task\_cmdb\_ci\_service |  |
| sys\_properties |  |
| kb\_knowledge |  |
| kb\_submission |  |
| interaction\_related\_record |  |
| metric\_definition |  |
| sysapproval\_approver |  |
| sysapproval\_group |  |
| contract\_sla |  |
| service\_offering\_commitment |  |
| on\_call\_escalation\_response |  |
| interaction |  |
| task\_cmdb\_ci\_service |  |

Objects

|  |  |
| --- | --- |
| SLAAsyncQueue |  |
| SLACalculator |  |
| Workflow() |  |
| current |  |
| TaskStateUtil |  |
| new global.VAGlobalUtil().isAttachmentCopyEnabled(current.getTableName()) |  |
| new InteractionRelationshipUtil().copyAttachments(interactionGr, current); |  |
| Workflow |  |
| sn\_major\_inc\_mgmt |  |
| GlideDateTime |  |
| TableUtils |  |
| GlideTransaction |  |
| GlidePluginManager |  |
| GlideSession |  |
| InteractionRelationshipUtil |  |
| GlideScriptRecordUtil |  |
| g\_scratchpad |  |
| JSON |  |
| global.OnCallRotation() |  |
| GlideScriptRecordUtil |  |
| GlideDBObjectManager |  |

Sys Properties

|  |  |
| --- | --- |
| glide.ui.autoclose.time | Inc auto close time |
| sn\_major\_inc\_mgmt.major\_incident\_management\_group |  |
|  |  |
|  |  |
|  |  |

Functions

|  |  |
| --- | --- |
| task\_sla.setWorkflow(false); |  |
| current.incident\_state.changesTo |  |
| var link = '<a href="' + current.getLink() + '">' + current.getDisplayValue() + '</a>'; |  |
| current.close\_notes.nil |  |
| current.isValidField("close\_code") | It must be returning false because of read acl on that column for that end user.  login with your end user and do table.list in left nav it should open list of records and then in the gear icon try to search that field.  If that field is not shown then it means user cannot access this column and hence isValidField() returning as false. |
| current.incident\_state.changesTo(IncidentState.RESOLVED) |  |
| **isNewRecord()** validates true if its validated \_before\_ insert into the db.  **isValidRecord()** validates if the record exists in the table (gr) - this can evaluate false even though a given record is not new (is inserted) |  |
| **workflow.cancel(current);** |  |
| **new InteractionRelationshipUtil().copyAttachments(interactionGr, current);** |  |
| **new global.VAGlobalUtil().isAttachmentCopyEnabled(current.getTableName()** |  |
| **sn\_major\_inc\_mgmt.ConsolidatedIncidentTimeline().getTimeline(current);** |  |
| **GlidePluginManager.isActive('sn\_fsm\_itsm\_mng')** |  |
| **getNextObjNumberPadded();** |  |
| **g\_scratchpad.incidentState = new JSON().encode(IncidentState);** |  |
| **Workflow().broadcastEvent(escRespGr.wf\_context, 'resume');** |  |
| **var cntx = new Workflow().startFlow(workflow, current, null, null);** |  |
| **current.work\_notes.setJournalEntry** |  |
| **var gr = new GlideRecord(current.parent.getTableName());** |  |

Gs

|  |  |
| --- | --- |
| gs.eventQueue("incident.commented", current, gs.getUserID(), gs.getUserName()); |  |
| gs.workflowFlush(current); |  |
| gs.hasRole |  |
| gs.addInfoMessage |  |
| gs.getMessage |  |
| gs.calDateDiff |  |
| gs.dateDiff(opened, closed, false); |  |
| gs.nowDateTime(); |  |

Current & Glide RecordFunctions

|  |  |
| --- | --- |
| current.sys\_id |  |
| current.incident\_state.changesTo |  |
| var link = '<a href="' + current.getLink() + '">' + current.getDisplayValue() + '</a>'; |  |
| current.close\_notes.nil |  |
| current.isValidField("close\_code") |  |
| current.incident\_state.changesTo(IncidentState.RESOLVED) |  |
| **isNewRecord()** validates true if its validated \_before\_ insert into the db.  **isValidRecord()** validates if the record exists in the table (gr) - this can evaluate false even though a given record is not new (is inserted) |  |
| **current.getDisplayValue** |  |
| **current.getValue("major\_incident\_state")** |  |
| **current.setValue("major\_incident\_state", "");** |  |
| **current.setAbortAction(true);** |  |
| **getTableName()** |  |
| getValue()  getDisplayValue() |  |
| GlideRecord(String tableName) | Creates an instance of the GlideRecord class for the specified table. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| addActiveQuery() | Adds a filter to return active records. |
| addEncodedQuery(String query) | Adds an encoded query to other queries that may have been set.  Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.  You can set the glide.invalid\_query.returns\_no\_rows system property to true to have queries with invalid encoded queries return no records. |
| addFunction(Object function) | Applies a pre-defined GlideDBFunctionBuilder object to a record.  Use the GlideDBFunctionBuilder scoped class to define a function. After the function is defined, use the addFunction(Object function) method to apply the function to a record.  var functionBuilder = new GlideDBFunctionBuilder();  var myAddingFunction = functionBuilder.add();  myAddingFunction = functionBuilder.field('order');  myAddingFunction = functionBuilder.field('priority');  myAddingFunction = functionBuilder.build();  // Query incidents in which order + priority is less than 5 and display that added value  var now\_GR = new GlideRecord('incident');  now\_GR.addFunction(myAddingFunction);  now\_GR.addQuery(myAddingFunction, '<', 5);  now\_GR.query();  while(now\_GR.next())  gs.info(now\_GR.getValue(myAddingFunction)); |
| addNotNullQuery(String fieldName) | A filter that specifies records where the value of the field passed in the parameter is not null. |
| addNullQuery(String fieldName) | Adds a filter to return records where the value of the specified field is null. |
| addQuery(String name, Object value) | Build a search query and return the rows that match the request. |
| addQuery(String name, String operator, Object value) | Provides the ability to build a request, which when executed, returns the rows from the specified table, that match the request. |
| addValue(String field, Number value) | Provides atomic add and subtract operations on a specified number field at the database level for the current GlideRecord object. |
| applyEncodedQuery(String queryString) | Sets the values of the specified encoded query terms and applies them to the current GlideRecord. |
| canCreate() | Determines if the Access Control Rules, which include the user's roles, permit inserting new records in this table. |
| canDelete() | Determines if the Access Control Rules, which include the user's roles, permit deleting records in this table. |
| canRead() | Determines if the Access Control Rules (ACLs) permit reading records in this table. This method evaluates all ACL types, such as user roles, scripted ACLs, ACLs with scripted conditions, and so on. |
| canWrite() | Determines if the Access Control Rules, which include the user's roles, permit editing records in this table. |
| chooseWindow(Number firstRow, Number lastRow, Boolean forceCount) | Sets a range of rows to be returned by subsequent queries. |
| deleteMultiple() | Deletes all records that satisfy the query.  This method does not delete attachments.  Do not use deleteMultiple() on tables with currency fields. Always delete each record individually. Also, do not use this method with the chooseWindow() or setLimit() methods when working with large tables. The setLimit() method does not limit the number of records that are deleted with deleteMultiple(). All records returned by the query are deleted regardless of setLimit(). |
| deleteRecord() | Deletes the current record. |
| get(Object name, Object value) | Returns the specified record in the current GlideRecord object.  This method accepts either one or two parameters. If only a single parameter is passed in, the method assumes that it is the sys\_id of the desired record. If not found, it then tries to match the value against the display value. If two parameters are passed in, the first is the name of the column within the GlideRecord to search. The second is the value to search for.  If multiple records are found, use next() to access the additional records. |
| current.operation() != 'insert' |  |
| current.addQuery("caller\_id", u).addOrCondition("opened\_by", u).addOrCondition("watch\_list", "CONTAINS", u); | Add additional qualifications for the Query |
| current.number = getNextObjNumberPadded(); |  |
| current.isNewRecord |  |
| current.sys\_class\_name |  |
| current.setAbortAction(true);  current.parent\_incident.setError(gs.getMessage('Invalid Parent Incident')); |  |

addJoinQuery(String joinTable, String primaryField, String joinTableField)

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

You can use this method to find all the users that are in the database group via the Group Member [sys\_user\_grmember] table, or to find all problems that have an assigned incident via the incident.problem\_id relationship.

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.)

Always test queries on a sub-production instance prior to deploying them on a production instance. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table. Using an insert(), update(), deleteRecord(), or deleteMultiple() method on bad query results can result in data loss.

You can set the glide.invalid\_query.returns\_no\_rows system property to true to have queries with invalid encoded queries return no records.

Parameters

Name Type Description

joinTable String Name of table to use in the join, such as 'incident'.

primaryField String Optional. Name of the field in the GlideRecord to use to join the field specified in the joinTableField parameter.

Default: sys\_id

joinTableField String Optional. Name of the field in the table specified in joinTable to use to join the tables.

Default: First field in the table specified in joinTable that is a reference field to the current GlideRecord table.

Returns

Type Description

GlideQueryCondition Filter that lists records where the relationships match.

Example

Find problems that have an incident attached. This 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 inactive problems with associated incidents.

// Look for Problem records that have associated Incident records

var now\_GR = new GlideRecord('problem');

var grSQ = now\_GR.addJoinQuery('incident');

// Where the Problem records are "active=false"

now\_GR.addQuery('active', 'false');

// And the Incident records are "active=true"

grSQ.addCondition('active', 'true');

// Query

now\_GR.query();

// Iterate and output results

while (now\_GR.next()) {

gs.info(now\_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 now\_GR = new GlideRecord('problem');

now\_GR.addJoinQuery('incident', 'opened\_by', 'caller\_id');

now\_GR.query();

Gs functions

|  |  |
| --- | --- |
| gs.include('DurationCalculator'); |  |
| gs.addInfoMessage(gs.getMessage('{0} is no longer a Major Incident Candidate', current.getDisplayValue())); |  |
| gs.getSession() |  |
| session.getClientData('parent\_interaction'); |  |
| session.clearClientData('parent\_interaction'); |  |
| gs.nil(interactionId) |  |

Glide Session Functions

|  |  |
| --- | --- |
| getClientData(String paramName) | Retrieves a session client value previously set with putClientData().  Use this method to retrieve data values that were set using the putClientData() method. |
| getClientIP() | Returns the client IP address. |
| getCurrentApplicationId() | Returns the application currently selected in the application picker.  This method requires admin privileges |
| getCurrentDomainID() | Returns the sys\_id of the current domain for the logged-in user session.  The identifier that is returned depends on the domain type and the instantiation of that domain.  If the user is configured in the global domain, and does not use the domain picker to switch domains, the method returns null.  If the user uses the domain picker to switch to the global domain, the method returns the string "global".  For all other domains, the method returns the sys\_id of that domain. |
| getLanguage() | Returns the session's language code. |
| getSessionToken() | Returns the session token. |
| getTimeZoneName() | Returns the name of the session's time zone. |
| getUrlOnStack() | Returns the URL on the stack. Returns null if the stack is empty. |
| isImpersonating() | Returns true if the user is impersonating another user. |
| isInteractive() | Returns true if the session is interactive.  An interactive session is one that involves an end-user interacting with a user interface that then retrieves information from a server. An example of this type of session is when a user logs in using the log-in screen or uses a form to query a data store. A non-interactive session is one that only involves programmatic interaction with a server such as a SOAP request to retrieve data. |
| isLoggedIn() | Returns true if the user is logged in. |
| putClientData(String paramName, String paramValue) | Sets a session client value that can be retrieved with getClientData(). This method is used in a server side script that runs when a form is created. |
|  |  |

GlideSysAttachment - Global

Release version:

Utah

UpdatedFeb 1, 20233 minutes to read

UtahAPI reference

The GlideSysAttachment API provides a way to handle attachments.

Content is returned as a string, not as a byte array when getContent() is called.

Content is returned as a GlideScriptableInputStream object when getContentStream() is called. The GlideScriptableInputStream contains the actual bytes not converted into a String.

GlideSysAttachment - copy(String sourceTable, String sourceID, String targetTable, String targetID)

Copies attachments from the source record to the target record.

Parameters

Name Type Description

sourceTable String Name of the table with the attachments to be copied.

sourceID String Source table's sys\_id.

targetTable String Name of the table on which to add the attachments.

targetID String Target table's sys\_id.

Returns

Type Description

String Array of sys\_ids of the attachments that were copied.

Example

var attachment = new GlideSysAttachment();

var incidentSysID = 'ab1b30031b04ec101363ff37dc4bcbfc';

var incGR = new GlideRecord('incident');

incGR.get(incidentSysID);

var copiedAttachments = attachment.copy('incident', incidentSysID, 'problem', incGR.getValue('problem\_id'));

gs.info('Copied attachments: ' + copiedAttachments);

Output:

Copied attachments: 6e4621df1bc420501363ff37dc4bcbb0,a87769531b0820501363ff37dc4bcba2

GlideSysAttachment - deleteAttachment(String attachmentID)

Deletes the specified attachment.

Parameters

Name Type Description

attachmentID String Attachment's sys\_id.

Returns

Type Description

void

Example

var attachment = new GlideSysAttachment();

var attachmentSysID = 'a87769531b0820501363ff37dc4bcba2';

attachment.deleteAttachment(attachmentSysID);

GlideSysAttachment - getAttachments(String tableName, String sys\_id)

Returns a GlideRecord containing the matching attachment metadata such as name, type, or size.

Parameters

Name Type Description

tableName String Name of the table to which the attachment belongs; for example, incident.

sys\_id String The sys\_id of record to which the attachment belongs.

Returns

Type Description

GlideRecord GlideRecord object containing the matching attachment metadata such as name, type, or size.

Example

The following script lists attachment file names for a record with two attachments.

var attachment = new GlideSysAttachment();

var agr = attachment.getAttachments('<table\_name>', '<record\_sys\_id>');

while(agr.next())

gs.info(agr.getValue('file\_name'));

Output:

\*\*\* Script: filename1.txt

\*\*\* Script: filename2.txt

GlideSysAttachment - getContentStream(String sysID)

Returns a GlideScriptableInputStream object given the sys\_id of an attachment.

You can use the GlideTextReader API to read the content stream.

Parameters

Name Type Description

sysID String Attachment sys\_id.

Returns

Type Description

GlideScriptableInputStream Stream that contains the attachment content.

Example

var attachment = new GlideSysAttachment();

var attachmentSysID = '6e4621df1bc420501363ff37dc4bcbb0';

var attachmentContentStream = attachment.getContentStream(attachmentSysID);

gs.info('Attachment content stream: ' + attachmentContentStream);

Output:

Attachment content stream: com.glide.communications.GlideScriptableInputStream@14bd299

GlideSysAttachment - GlideSysAttachment()

Creates an instance of the GlideSysAttachment class.

Parameters

Name Type Description

None

GlideSysAttachment - write(GlideRecord record, String fileName, String contentType, String content)

Attaches a specified attachment to the specified record.

Parameters

Name Type Description

record GlideRecord Record to which to attach the attachment.

fileName String Attachment file name.

contentType String Attachment content type.

content String Attachment content.

Returns

Type Description

String Attachment sys\_id. Returns null if the attachment was not added.

Example

var attachment = new GlideSysAttachment();

//set up inputs

var rec = new GlideRecord('incident');

rec.get('78271e1347c12200e0ef563dbb9a7109');

var fileName = 'example.txt';

var contentType = 'text/csv';

var content = 'The text that is stored inside my file';

var agr = attachment.write(rec, fileName, contentType, content);

gs.info('The attachment sys\_id is: ' + agr);

Output:

The attachment sys\_id is: 01271e4317c13311e0ef563dbb9abe34

GlideSysAttachment - writeContentStream(GlideRecord now\_GR, String fileName, String contentType, GlideScriptableInputStream inputStream)

Inserts an attachment using the input stream.

Parameters

Name Type Description

now\_GR GlideRecord Record to which the attachment is to be attached.

fileName String Attachment's file name.

contentType String Attachment's content type.

content GlideScriptableInputStream Attachment content.

Returns

Type Description

String Sys\_id of the attachment created.

Example

Attaches a content stream from the sys\_attachment table to a test\_table record.

function copyAttachmentToGlideRecord(conceptSysId) {

// Get record from test\_table using sys\_id

var targetGlideRecord = new GlideRecord("test\_table");

if (!targetGlideRecord.get(conceptSysId)) {

throw ("Cannot find record created by test with sys\_id: " + conceptSysId);

}

// Get record from sys\_attachment table

var sourceAttachmentGlideRecord = new GlideRecord('sys\_attachment');

sourceAttachmentGlideRecord.query();

sourceAttachmentGlideRecord.next();

// Get field values from retrieved sys\_attachment record

var fileName = sourceAttachmentGlideRecord.getValue('file\_name');

var contentType = sourceAttachmentGlideRecord.getValue('content\_type');

var sourceAttachmentSysId = sourceAttachmentGlideRecord.getValue('sys\_id');

// Attach sys\_attachment record content stream to test\_table record

var gsa = new GlideSysAttachment();

gsa.writeContentStream(

targetGlideRecord,

fileName,

contentType,

gsa.getContentStream(sourceAttachmentSysId));

gs.info("Attachment created");

}

function isConnect() {

var transaction = GlideTransaction.get();

if (!transaction)

return false;

var request = transaction.getRequest();

if (!request)

return false;

var path = request.getRequestURI();

return path && path.match(/\/api\/now\/connect/);

}

Condition: new GlideappLiveFeedEventHandler().watchedTable(current.getTableName());

var gru = new GlideScriptRecordUtil.get(current)

var fieldsChanged = gru.getChangedFieldNames();

gs.eventQueue('live\_feed.update', current, fieldsChanged.toString(), current.sys\_mod\_count);