knowledge16

May 15–20, 2016 • Mandalay Bay • Las Vegas, NV

Lab Guide

Advanced GlideRecord Scripting

Steven Bell & Mark Amann

Default Login / Password:

admin / Knowledge16

itil / Knowledge16

employee / Knowledge16

servicenow

This

Page

Intentionally

Left

Blank

Introduction

The purpose of these labs to give the student a more detailed idea of what GlideRecords are capable of, and some of the best practices surrounding ServiceNow Scripting, and data retrieval.

What will be covered:

- **Lab 1.1 - Create a New Update Set**
- **Lab 1.2 - Modify Run Fix Script UI Action**
- **Lab 1.3 - Refactoring a GlideRecord in a Loop**
- **Lab 1.4 - Dot-Stacking a GlideRecord Query**
- **Lab 1.5 - Advanced Encoded Queries**
- **Lab 1.6 - Extending GlideRecord - Implementing Between**
- **Lab 1.7 - Extending GlideRecord - Implementing .toJSON (Bonus)**

Lab Goal

Lab 1.1
Create a New Update Set

The purpose of this lab is to show a best practice: Whenever you go to make any modifications to the ServiceNow application you should always create an update set prior to doing so. This helps you keep track of what has been modified, and will assist in promoting the modifications to another instance.

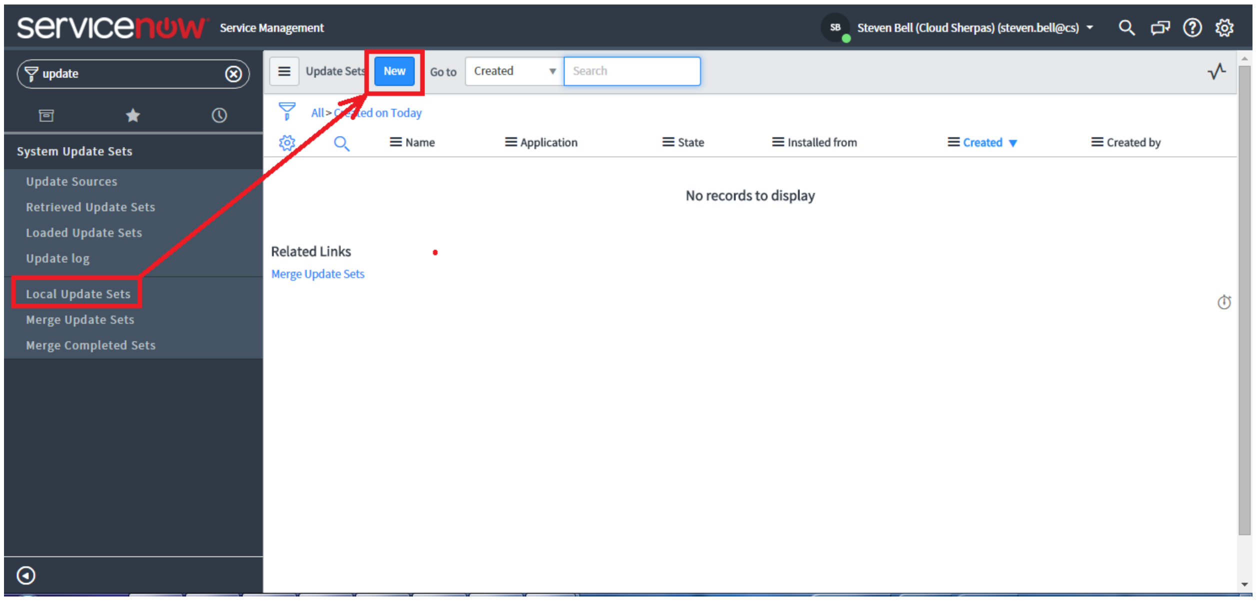
Creating a New Update Set

Log onto your training instance.

Type the following in the navigation search: **update**. The System Update Sets application will be displayed in the navigation menu.

Click the following module link: **Local Update Sets**. A list view of the local update sets will be displayed. A best practice is to drag this link over to the Edge Bar for easier later reference.

At the top of the List View click the **New** button. The create new Update Set form will be displayed.

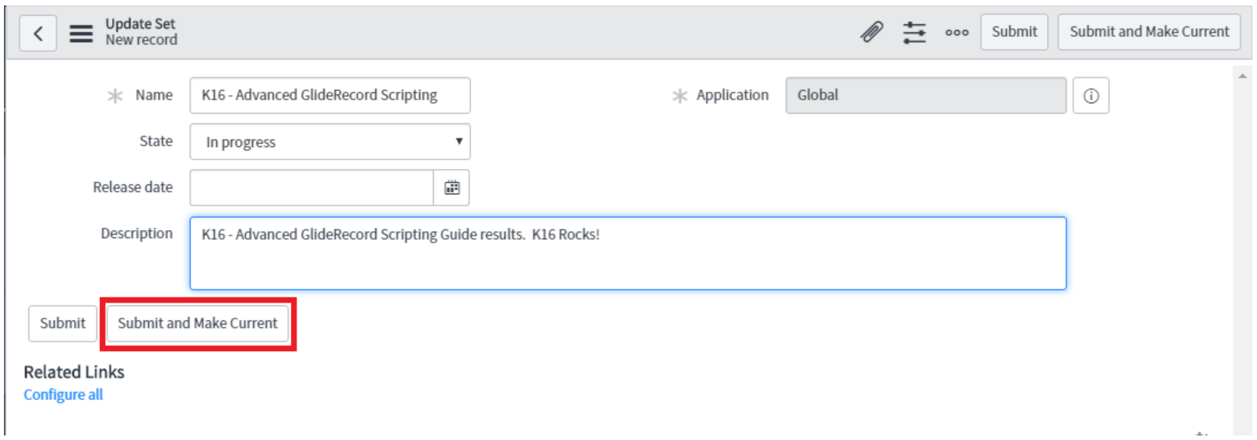


Fill in the following fields:

a. **Name**: Advanced GlideRecord Scripting Guide - <<your initials>> (or anything that you like)

b. **Description**: Advanced GlideRecord Scripting Guide results (again anything you like)

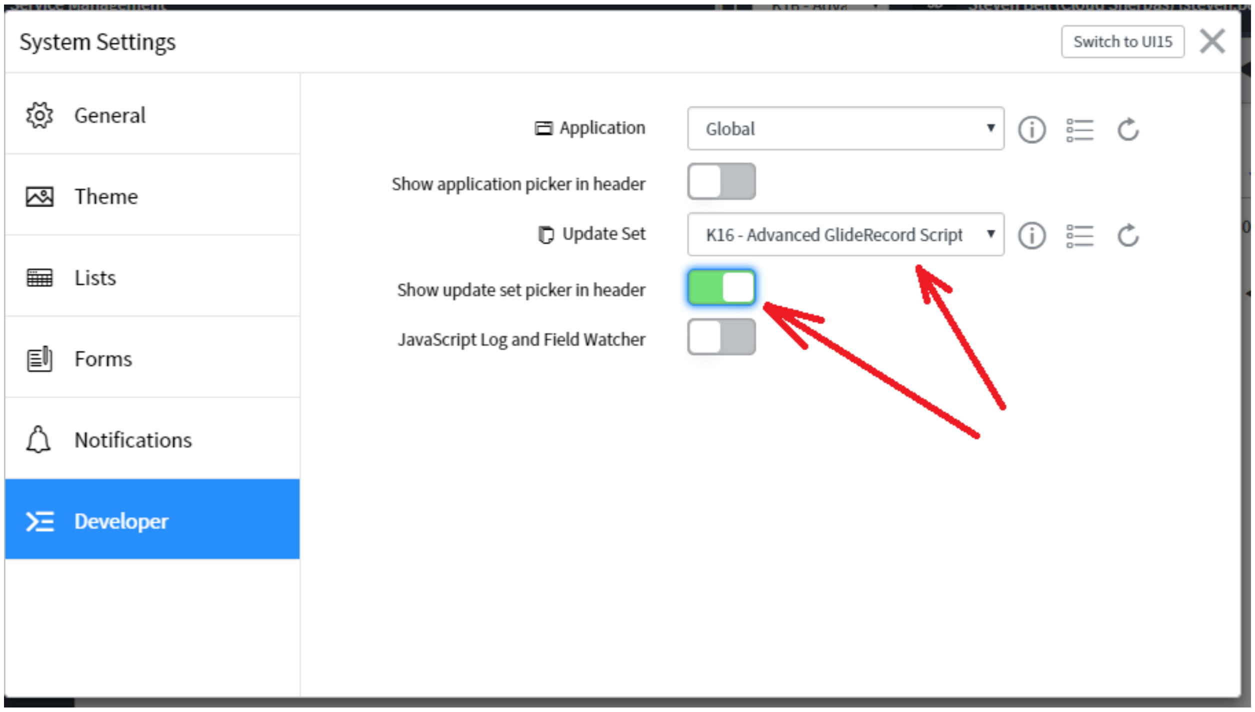
Click the **Submit And Make Current** button at the bottom of the form. This will create your new update set and make it the current one. You can verify this by clicking on the **Gear** button located in the upper left of your browser, and observing the current update set value.



Name	K16 - Advanced GlideRecord Scripting
Application	Global
State	In progress
Release date	
Description	K16 - Advanced GlideRecord Scripting Guide results. K16 Rocks!

You are done creating your update set!

Now one last step. You need to verify that the update set really is your current. Click on the **Gear** button in the upper right side. This opens the SeviceNow Configuration context menu. Scroll down and verify the current update set. It should look something like this:


System Settings

Switch to UI15

Application: Global

Show application picker in header: off

Update Set: K16 - Advanced GlideRecord Script

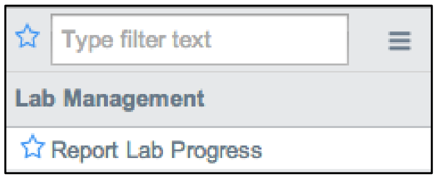
Show update set picker in header: on

JavaScript Log and Field Watcher: off

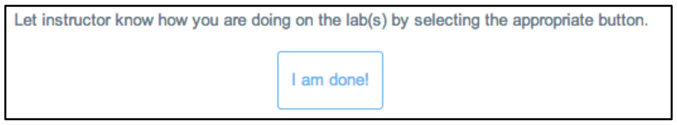
Don't forget to mark the update set complete and download it at the end of your session! If you need help be sure to ask one of the Teaching Assistants!

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.



Click **I am done!**


Let instructor know how you are doing on the lab(s) by selecting the appropriate button.

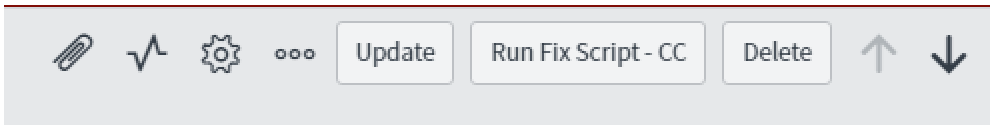
Lab Goal

Lab 1.2 Modify Run Fix Script UI Action

The purpose of this lab is to change the out-of-the-box location for this really useful UI Action to be available on the form as a button at the top. This is useful as after a "save" the form resets to the top of the screen, and you have to scroll down to find the Run Fix Script link in the related links.

Modifying The Run Fix Script UI Action

1. Navigate to System UI -> UI Actions. The UI Actions List View will be displayed.
2. Search for the Run Fix Script UI Action in the list and open the form.
3. Change the fields on the form to the following:
    a. Name: Run Fix Script – CC
    b. Form Button: Checked
    c. Form Context Menu: Checked
    d. List Context Menu: Checked
    e. Comments: Runs the current fix script. CreatorCon modification.
4. Right-click on the form header to bring up the form context menu, and choose Insert and Stay. This keeps you from modifying the out-of-the-box (and owning it).
5. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
6. Click on any of those displayed. You will now observe a button labeled: Run Fix Script – CC.



This button will be very useful when doing the labs.

Lab Goal

Lab 1.3 Refactoring a GlideRecord in a Loop

The purpose of this lab is to demonstrate:

- The bad practice of using a GlideRecord inside of a GlideRecord loop
- The best practice to use JavaScript arrays to refactor such a scenario down to two database calls
- The usage of the built-in ArrayUtil function “unique” to remove duplicates from an array
- Using the GlideRecord keyword “IN” to do array comparisons
- Refactoring code to improve efficiency

Problem Description

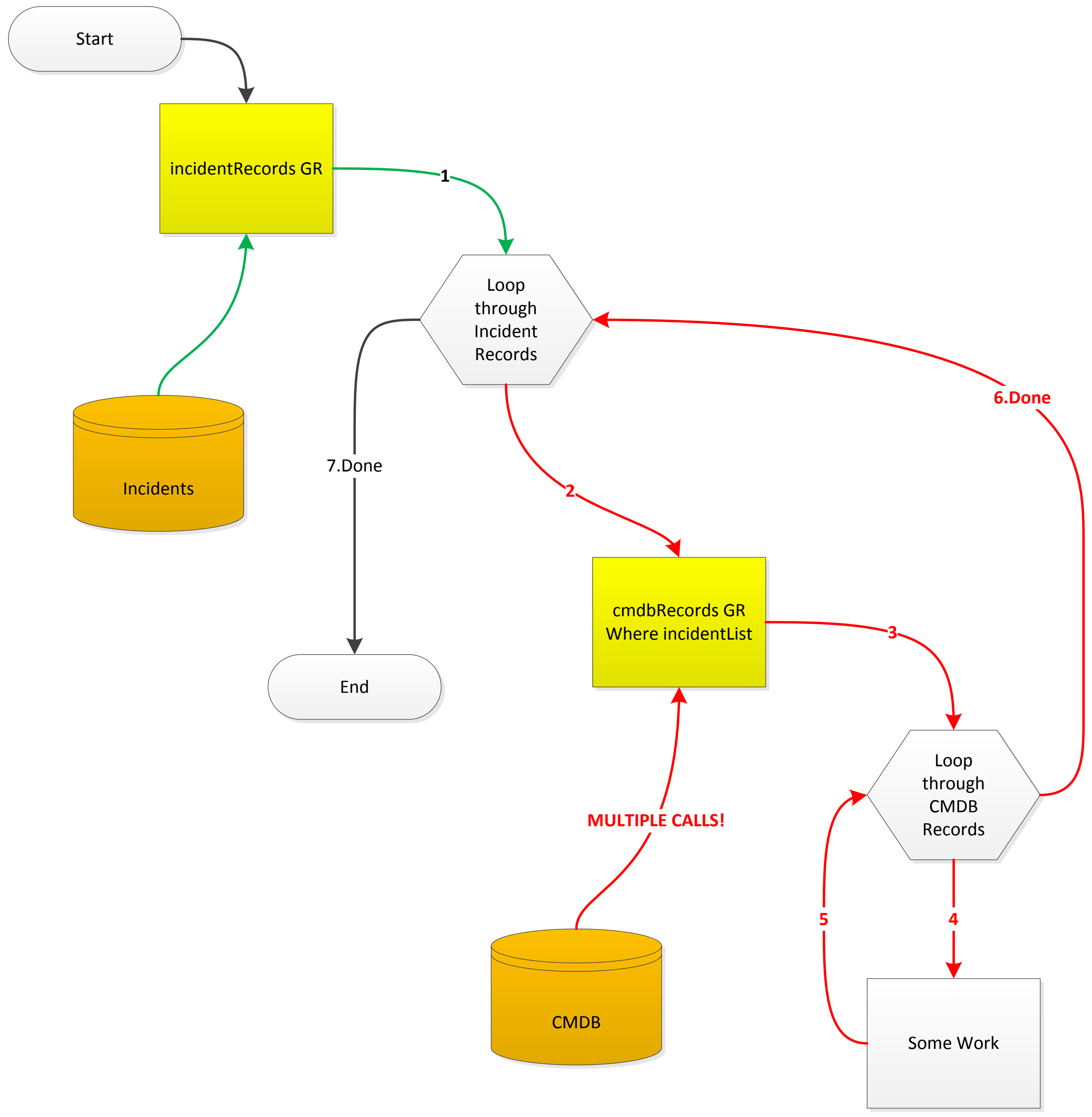
Often it is necessary to compare values from one table against values in another, and then do work with both. The common practice is to do a GlideRecord call to the first table, and then in the While loop do calls to the second table to do the comparison. If there are a lot of records this takes the server more time to execute because of multiple calls to the second table.

The amount of time it takes if done in a business rule could likely impact the user experience! Something NOT desired.

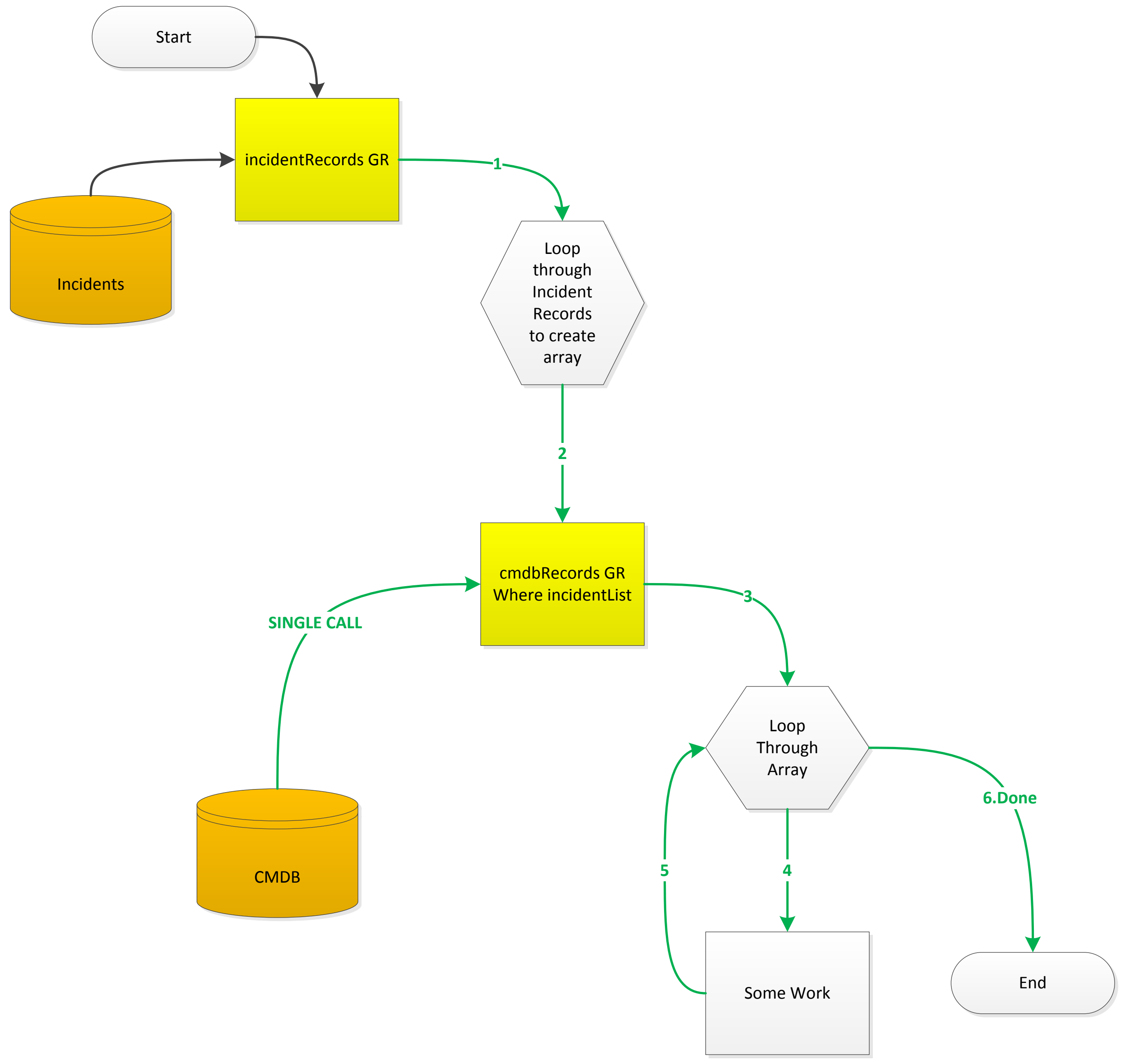
Also, with the creation of a GlideRecord object, the connection with the database remains open for the entirety of execution (even after you do an update, insert, or delete).

There is a way to improve performance by reducing the number of calls to a bare minimum; usually a grand total of two!

Before:


Start
incidentRecords GR
1
Loop through Incident Records
Incidents
7.Done
6.Done
2
End
cmdbRecords GR Where incidentList
3
Loop through CMDB Records
MULTIPLE CALLS!
5
4
CMDB
Some Work

After:


Start
incidentRecords GR
Incidents
1
Loop through Incident Records to create array
2
cmdbRecords GR Where incidentList
SINGLE CALL
CMDB
3
Loop Through Array
6.Done
5
4
Some Work
End

GlideRecord Inside-of-a-GlideRecord

First we will construct the query-inside-of-a-query-loop scenario, and do timings to have a baseline with which to measure our code performance.

1. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
2. Click on the New button. A new Fix Script form will be displayed.
3. Fill in the following fields:
    a. **Name**: CC Lab 1.3 GlideRecord Loop
    b. **Active**: checked
    c. **Run Once**: checked
    d. **Description**: GlideRecord Inside of a GlideRecord Loop Scenario
    e. **Script**:

```
// Example of a GlideRecord inside of a GlideRecord
var loggingSource = '\t****FS:CC Lab 1.3 glideRecordLoop';

// we won't limit the number of records because we want
// to execute a few to get a better timing
var incidents = new GlideRecord('incident');
incidents.orderBy('number');
incidents.query();

while (incidents.next()) {
    var cmdb_ci = new GlideRecord('cmdb_ci');
    cmdb_ci.addQuery('sys_id', incidents.cmdb_ci);
    cmdb_ci.query();

    while (cmdb_ci.next()) {
          gs.info('---> {0} {1}',
                        cmdb_ci.name,
                        loggingSource);
    }
}
```

    f. Right-click on the form header to bring up the context menu and choose Save to save your work.

4. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

5. Your results should look something like the following:

```
*** Script: ---> MailServerUS        ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> FileServerFloor2    ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> ApplicationServerPeopleSoft  ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> eFax       ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Saints and Sinners Bingo     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> IBM-T42-DLG         ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Windows XP Hotfix (SP2) Q817606       ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> WeatherBug          ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> CALLXPR1   ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> DatabaseServer2     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> EFOWEB     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Adobe Acrobat 7.0.3 and Reader 7.0.3 Update     ****FS:CC Lab 1.3 glideRecordLoo
p
*** Script: ---> Access IBM          ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> WEBSERVER  ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Sales Force Automation       ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> *JEMPLOYEE-IBM      ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Sales Force Automation       ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> nyc rac nas200      ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> EXCH-SD-05          ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> SAP Controlling     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> SAP Financial Accounting     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> SAP Human Resources ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> SAP Materials Management     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> SAP Sales and Distribution   ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> win-z4v89fexkhn     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Kastle     ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Blackberry          ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Blackberry          ****FS:CC Lab 1.3 glideRecordLoop
*** Script: ---> Bond Trading        ****FS:CC Lab 1.3 glideRecordLoop
[0:00:00.288] Total Time
```

Notice the duplicates? This is because there are CIs with more than one incident. Yet another draw-back to this method!

6. Observe the total time displayed and record here: ___________

Refactoring the GlideRecord Loop Structure

Now let's construct a query where the second GlideRecord call has been pulled out of the loop, and turned into an array of values.

7. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
8. Click on the New button. A new Fix Script form will be displayed.
9. Fill in the following fields:
    a. **Name**: CC Lab 1.3 GlideRecord Loop
    b. **Active**: checked
    c. **Run Once**: checked
    d. **Description**: GlideRecord Inside of a GlideRecord Loop Scenario

e. **Script**:

```
var loggingSource = '\t****FS:CC Lab 1.3 glideRecordLoopRefactored';

// First construct our comparison list
// Remember a one-dimensional array is a comma delimited list of
// values in JavaScript
var incidents = new GlideRecord('incident');
incidents.orderBy('number');
incidents.query();

// push the resultant sys_id recordset to a 1-dimensional array
incidentsList = [];
while (incidents.next()) {
    incidentsList.push(incidents.cmdb_ci + '');
}

// remove any duplicates (to minimize our number of loops further)
incidentsList = new ArrayUtil().unique(incidentsList);

// Now do our query with our array using the 'IN' statement
var cmdb_ci = new GlideRecord('cmdb_ci');
cmdb_ci.addQuery('sys_id', 'IN', incidentsList);
cmdb_ci.orderBy('name'); // a bonus! we can now order the list!
cmdb_ci.query();

while (cmdb_ci.next()) {
    gs.info('---> {0} {1}',
                cmdb_ci.name,
                loggingSource);
}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

10. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

11. Your results should look something like the following:

```
*** Script: ---> *JEMPLOYEE-IBM   ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Access IBM       ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Adobe Acrobat 7.0.3 and Reader 7.0.3 Update ****FS:CC Lab 1.3 glideRecordLoo
pRefactored
*** Script: ---> ApplicationServerPeopleSoft        ****FS:CC Lab 1.3 glideRecordLoopRefactor
ed
*** Script: ---> Blackberry       ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Bond Trading     ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> CALLXPR1         ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> DatabaseServer2  ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> eFax   ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> EFOWEB ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> EXCH-SD-05       ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> FileServerFloor2         ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> IBM-T42-DLG      ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Kastle ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> MailServerUS     ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> nyc rac nas200   ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Saints and Sinners Bingo  ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Sales Force Automation    ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> SAP Controlling  ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> SAP Financial Accounting  ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> SAP Human Resources       ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> SAP Materials Management  ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> SAP Sales and Distribution         ****FS:CC Lab 1.3 glideRecordLoopRefactor
ed
*** Script: ---> WeatherBug       ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> WEBSERVER        ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> win-z4v89fexkhn  ****FS:CC Lab 1.3 glideRecordLoopRefactored
*** Script: ---> Windows XP Hotfix (SP2) Q817606    ****FS:CC Lab 1.3 glideRecordLoopRefactor
ed
[0:00:00.060] Total Time
```

12. Observe the total time displayed and record here: ___________

How do the two times compare? Amazing huh?

My Results

- Before timing example: 0.288 seconds
- After timing example: 0.060 seconds

0.288 / 0.060 = 4.8

A factor of almost **five** time faster! Your results may vary, but it should show a marked improvement.

Progress Report

1. Navigate to Lab Management > Report Lab Progress.

   Click I am done!

Lab Goal

Lab 1.4 Dot-Stacking a GlideRecord Query

The purpose of this lab is to demonstrate:

- That complex queries can often be simplified
- Simplification of queries can be done using dot stacking
- Dot stacking is beneficial to maintenance
- Dot stacking is beneficial to understanding a complex query

Problem Description

Large involved queries can be difficult to maintain. Even with good commenting practices it may be difficult to understand what the query is actually doing.

An example of a complex SQL query:

```
SELECT * FROM <table>
WHERE <field> LIKE <value%> OR <field> LIKE <value2%>
AND <field2> LIKE <value3%> OR <field2> LIKE <value4%>
AND <field3> LIKE <value5%> OR <field3> LIKE <value6%>
```

This lab will demonstrate refactoring a query a number of times to reduce the complexity, and improve the readability and maintainability of the GlideRecord Query.

Typical OR GlideRecord

First write up the query with the common usage of GlideRecord “OR”.

1. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
2. Click on the New button. A new Fix Script form will be displayed.
3. Fill in the following fields:
    a. **Name**: CC Lab 1.4 Typical OR GlideRecord
    b. **Active**: checked
    c. **Run Once**: checked
    d. **Description**: Typical OR Condition GlideRecord

e. **Script**:

```
var loggingSource = '\t****FS:CC Lab 1.4 typicalORQuery';

var incidentRecords = new GlideRecord('incident');
incidentRecords.addActiveQuery();  // make sure the record is active==true

// AND in our first query and then tack in the other OR's
var incQuery = incidentRecords.addQuery('location.name', 'CONTAINS', 'San Diego');
incQuery.addOrCondition('location.name', 'CONTAINS', 'Salt Lake');
incQuery.addOrCondition('location.name', 'CONTAINS', 'New York');

// AND in our second query, and tack on the OR's
var incQuery2 = incidentRecords.addQuery('priority', 1);
incQuery2.addOrCondition('priority', 3);

// AND in our third query, and tack on the OR's
var incQuery3 = incidentRecords.addQuery('state', 2);
incQuery3.addOrCondition('state', 4);

incidentRecords.orderBy('number');
incidentRecords.query();

// now display the results
while (incidentRecords.next()) {
    // we can only have up to five in a row
    // then we have to goto an array of values
    gs.info('---> Number: {0}, {1}, {2}, {3} {4}',
        incidentRecords.number,
        incidentRecords.priority,
        incidentRecords.state,
        incidentRecords.location.name,
          loggingSource);
}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

4. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

5. Your results should look something like the following:

```
*** Script: ---> Number: INC0000003, 1, 2, Salt Lake City    ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000007, 1, 4, San Diego         ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000015, 1, 2, Salt Lake City    ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000025, 1, 2, Salt Lake City    ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000046, 3, 2, Salt Lake City    ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000047, 3, 2, San Diego         ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000050, 1, 2, San Diego         ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000051, 1, 2, San Diego         ****FS:CC Lab 1.4 typicalORQuery
-
*** Script: ---> Number: INC0000053, 1, 2, 937 Lomas Santa Fe Drive, San Diego, CA

****FS:CC Lab 1.4 typicalORQuery
[0:00:00.027] Total Time
```

Solution – Dot Notation

Now let us simplify this using “dot” notation instead!

6. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
7. Click on the New button. A new Fix Script form will be displayed.
8. Fill in the following fields:
    a. **Name**: CC Lab 1.4 Dot Walked OR GlideRecord
    b. **Active**: checked
    c. **Run Once**: checked
    d. **Description**: Dot Walked OR Condition GlideRecord

e. **Script**:

```
var loggingSource = '\t****FS:CC Lab 1.4 dotWalkedORQuery';

var incidentRecords = new GlideRecord('incident');
incidentRecords.addActiveQuery();  // make sure the record is active==true

// notice that this cleans up everything and makes it MUCH easier to maintain!
incidentRecords.addQuery('location.name', 'CONTAINS', 'San Diego')
    .addOrCondition('location.name', 'CONTAINS', 'Salt Lake')
    .addOrCondition('location.name', 'CONTAINS', 'New York');
incidentRecords.addQuery('priority', 1).addOrCondition('priority', 3);
incidentRecords.addQuery('state', 2).addOrCondition('state', 4);

incidentRecords.orderBy('number');
incidentRecords.query();

// now display the results
while (incidentRecords.next()) {
    // we can only have up to five in a row
    // then we have to goto an array of values
     gs.info('---> Number: {0}, {1}, {2}, {3} {4}',
         incidentRecords.number,
         incidentRecords.priority,
         incidentRecords.state,
         incidentRecords.location.name,
           loggingSource);
}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

9. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

10. Your results should look exactly the same as before!

NOTE: CONTAINS is the same as LIKE, but if you use LIKE you will have to add a percent to the front and back end of your target string. For example:

incidentRecords.addQuery('location.name', 'LIKE', '%San Diego%');

Dot Notation Refactored… Slightly

We can reduce this a bit more by consolidating some OR condition statements into a single “IN” query! This really only can work though if we have exact values we are looking for. CONTAINS and LIKE do not covert into an “IN” query. The IN statement looks through a list of comma delimited values for a match. You can have as many values in the list as you want. As a matter-of-fact you can substitue a single dimension array into the list if needed.

11. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
12. Click on the New button. A new Fix Script form will be displayed.
13. Fill in the following fields:
    a. **Name**: CC Lab 1.4 Refactored Dot Walked OR
    b. **Active**: checked
    c. **Run Once**: checked
    d. **Description**: Refactored Dot Walked OR Condition GlideRecord

e. **Script**:

```
var loggingSource = '\t****FS:CC Lab 1.4 refactoredDotWalkedORQuery';

// this is to demonstrate that an array is a comma-delimited list
var priorities = [1,3];

var incidentRecords = new GlideRecord('incident');
incidentRecords.addActiveQuery();  // make sure the record is active==true

// Query refactored to get rid of the .addOrCondition for exact values
incidentRecords.addQuery('location.name', 'CONTAINS', 'San Diego')
    .addOrCondition('location.name', 'CONTAINS', 'Salt Lake')
    .addOrCondition('location.name', 'CONTAINS', 'New York');
incidentRecords.addQuery('priority', 'IN', priorities);
incidentRecords.addQuery('state', 'IN', '2,4');

incidentRecords.orderBy('number');
incidentRecords.query();

// now display the results
while (incidentRecords.next()) {
    // we can only have up to five in a row
    // then we have to goto an array of values
    gs.info('---> Number: {0}, {1}, {2}, {3} {4}',
        incidentRecords.number,
        incidentRecords.priority,
        incidentRecords.state,
        incidentRecords.location.name,
          loggingSource);
}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

14. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

15. Again, your results should look exactly the same as before!

Progress Report

1. Navigate to Lab Management > Report Lab Progress.

   Click I am done!

Lab Goal

The purpose of this lab is to demonstrate:

- Basic binary operations - the life-blood of a query
- Building an encoded query with a list view
- Simplification of complex GlideRecord queries
- The various functions available with encoded queries

Lab 1.5 Advanced Encoded Queries

Problem Description

Review of the state table for AND and OR evaluation:

	Operator		Result
TRUE	AND	TRUE	TRUE
TRUE	AND	FALSE	FALSE
FALSE	AND	TRUE	FALSE
FALSE	AND	FALSE	FALSE
TRUE	OR	TRUE	TRUE
TRUE	OR	FALSE	TRUE
FALSE	OR	TRUE	TRUE
FALSE	OR	FALSE	FALSE

The order of precedence for evaluation is always:

1. Inside the parenthesis
2. AND
3. OR

So, for example:

FALSE OR FALSE AND (TRUE OR FALSE)

Would evaluate to this as the parenthesis would evaluate first:

FALSE OR FALSE AND TRUE

Would evaluate to this as the AND would evaluate second:

FALSE OR FALSE

Would evaluate to this as the OR would be last to evaluate:

FALSE

All computer and database languages follow this paradigm including SQL.

Now the zinger: Parenthesis are NOT allowed in encoded queries! They are illegal and cause the GlideRecord encoded query to error, and therefore would be thrown out silently during execution. Be very careful how you set up your encoded query and how you want it to behave. You are limited in your set of tools.

Solution – Encoded Query

1. Just a note: we will be re-writing the query code from lab 1.4. We will be changing out the five lines containing our query code, and replacing them with an encoded query.

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

incidentRecords.addQuery('location.name', 'CONTAINS', 'San Diego')
    .addOrCondition('location.name', 'CONTAINS', 'Salt Lake')
    .addOrCondition('location.name', 'CONTAINS', 'New York');
incidentRecords.addQuery('priority', 'IN', '1, 3');
incidentRecords.addQuery('state', 'IN', '2, 4');

incidentRecords.orderBy('number');
incidentRecords.query();
```

2. Navigate to Incident -> All. The list view of all incidents will appear.

3. Click on the filter button and create the following query:

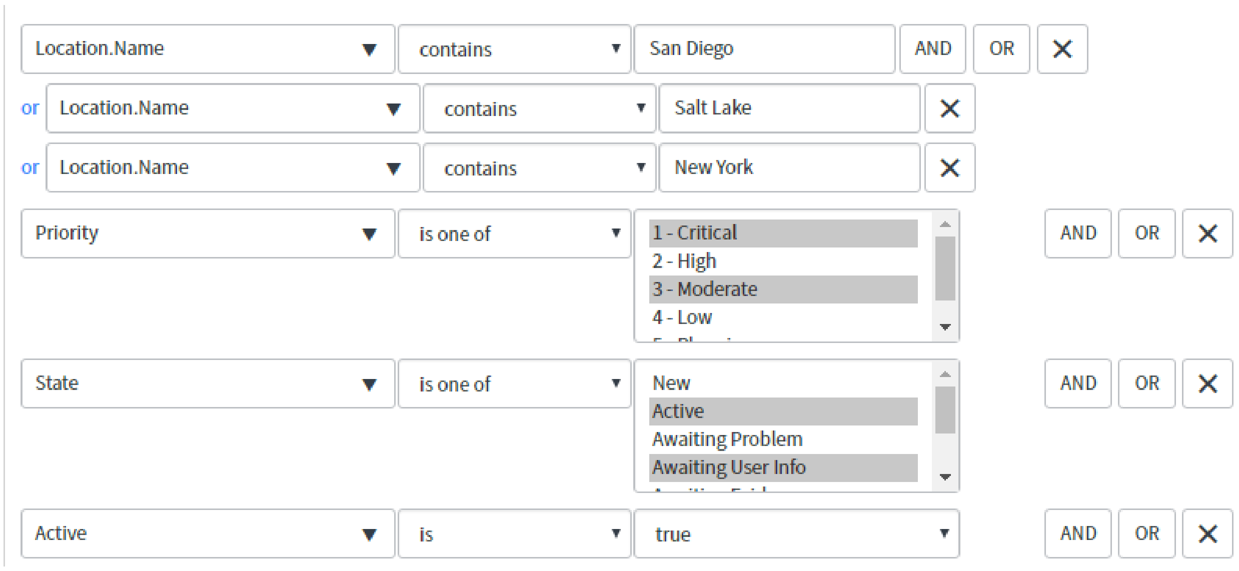
    a. Location.Name contains San Diego, or Salt Lake, or New York

    b. Priority is Critical or Moderate

    c. State is Active or Awaiting User Info

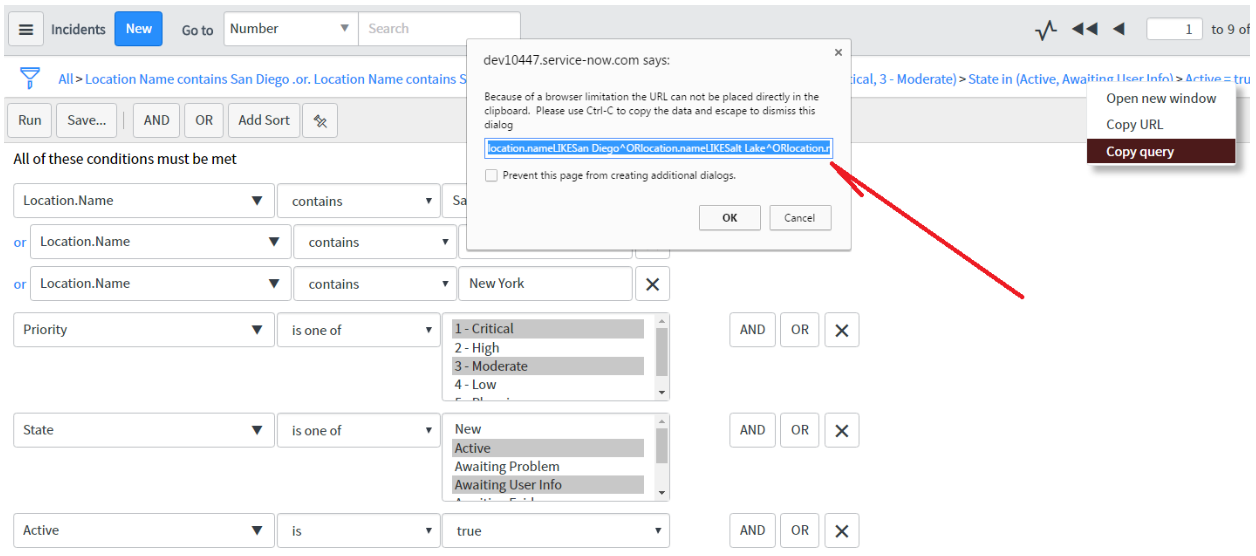
    d. Active is True

4. Your query should look something like this:



	Location.Name	contains	San Diego
or	Location.Name	contains	Salt Lake
or	Location.Name	contains	New York
	Priority	is one of	1 - Critical, 3 - Moderate
	State	is one of	Active, Awaiting User Info
	Active	is	true

5. Now right click on the very end of the breadcrumbs at the top (on the Active=true) part and choose Copy Query. Copy the query out of the pop-up box that appears.



All > Location Name contains San Diego .or. Location Name contains S … ical, 3 - Moderate) > State in (Active, Awaiting User Info) > Active = tru

All of these conditions must be met

dev10447.service-now.com says:

Because of a browser limitation the URL can not be placed directly in the clipboard. Please use Ctrl-C to copy the data and escape to dismiss this dialog

location.nameLIKESan Diego^ORlocation.nameLIKESalt Lake^ORlocation.r

Prevent this page from creating additional dialogs.

a. Paste this into your favorite note pad application (I use Notepad++ for example) for use a couple steps from now.

   Mine looked like this:

   ```
   location.nameLIKESan Diego^ORlocation.nameLIKESalt
   Lake^ORlocation.nameLIKENew York^priorityIN1,3^stateIN2,4^active=true
   ```

b. Note the number of records that are displayed here: _______

c. Look at the records and get an idea of the Incident Number, State, Location. This will give you some idea whether the code we test later is actually working.

6. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.

7. Click on the New button. A new Fix Script form will be displayed.

8. Fill in the following fields:

   a. **Name**: CC Lab 1.5 Encoded Query

   b. **Active**: checked

   c. **Run Once**: checked

   d. **Description**: Test the encoded query

e. **Script**:

```
var loggingSource = '\t****FS:CC Lab 1.5 encodedQuery-';

var sql = '<<paste your encoded query here!>>';

testEncodedQuery(sql, loggingSource);

function testEncodedQuery(sql, loggingSource) {
    var incidentRecords = new GlideRecord('incident');
    incidentRecords.addEncodedQuery(sql); // now add in our encoded query
    incidentRecords.orderBy('number');
    incidentRecords.query();

    while (incidentRecords.next()) {
          // note with the comma delimited gs.info you
         // can only have a maximum of five values
          gs.info('---> Number: {0}, {1}, {2}, {3} {4}',
                incidentRecords.number,
                incidentRecords.priority,
                incidentRecords.state,
                incidentRecords.location.name,
                loggingSource + 'testEncodedQuery');

    }

}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

9. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

10. Your results should look something like the following:

```
*** Script: ---> Number: INC0000003, 1, 2, Salt Lake City        ****FS:CC Lab 1.5 encodedQuery-t
estEncodedQuery
*** Script: ---> Number: INC0000007, 1, 4, San Diego    ****FS:CC Lab 1.5 encodedQuery-testEncode
dQuery
*** Script: ---> Number: INC0000015, 1, 2, Salt Lake City        ****FS:CC Lab 1.5 encodedQuery-t
estEncodedQuery
*** Script: ---> Number: INC0000025, 1, 2, Salt Lake City        ****FS:CC Lab 1.5 encodedQuery-t
estEncodedQuery
*** Script: ---> Number: INC0000046, 3, 2, Salt Lake City        ****FS:CC Lab 1.5 encodedQuery-t
estEncodedQuery
*** Script: ---> Number: INC0000047, 3, 2, San Diego    ****FS:CC Lab 1.5 encodedQuery-testEncode
dQuery
*** Script: ---> Number: INC0000050, 1, 2, San Diego    ****FS:CC Lab 1.5 encodedQuery-testEncode
dQuery
*** Script: ---> Number: INC0000051, 1, 2, San Diego    ****FS:CC Lab 1.5 encodedQuery-testEncode
dQuery
*** Script: ---> Number: INC0000053, 1, 2, 937 Lomas Santa Fe Drive, San Diego, CA ****FS:CC Lab
1.5 encodedQuery-testEncodedQuery
[0:00:00.028] Total Time
```

11. Write the number of records here: _________

Is this the number of records you were expecting?

Does the data look like what was displayed in by the List View query?

Refactored Encoded Query

So let's reorder and cleanup the code a bit (yes, it's legal!):

12. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.

13. Click on the New button. A new Fix Script form will be displayed.

14. Fill in the following fields:

    a. **Name**: CC Lab 1.5 Encoded Query Refactored

    b. **Active**: checked

    c. **Run Once**: true

    d. **Description**: Test the encoded query – refactored with a bunch of cool stuff addeded in!

e. **Script**:

```
var loggingSource = '\t****FS:CC Lab 1.5 encodedQuery refactored!-';

// makes it easier to read and maintain.  Your encoded query should look similar
var sql = 'location.nameLIKESan Diego' +
    '^ORlocation.nameLIKESalt Lake' +
    '^ORlocation.nameLIKENew York' +
    '^priorityIN1,3' +
    '^stateIN2,4' +
    '^active=true';

testEncodedQuery(sql, loggingSource);

function testEncodedQuery(sql, loggingSource) {
    var message = '\n'; // neat trick to clean up your output

    var incidentRecords = new GlideRecord('incident');
    incidentRecords.addEncodedQuery(sql); // now add in our encoded query
    incidentRecords.orderBy('number');
    incidentRecords.query();

    while (incidentRecords.next()) {
          // http://docs.oracle.com/javase/6/docs/api/?java/text/MessageFormat.html
          // you have to have the array of values, won't work otherwise!
          message += gs.getMessage('---> Number: {0}, {1}, {2}, {3}\n',
                [incidentRecords.number,
                incidentRecords.priority,
                incidentRecords.state,
                incidentRecords.location.name]);

    }

    gs.info('{0} {1}', message, loggingSource + 'testEncodedQuery');

}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

15. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

16. Your results should look something like the following:

```
*** Script:
---> Number: INC0000003, 1, 2, Salt Lake City
---> Number: INC0000007, 1, 4, San Diego
---> Number: INC0000015, 1, 2, Salt Lake City
---> Number: INC0000025, 1, 2, Salt Lake City
---> Number: INC0000046, 3, 2, Salt Lake City
---> Number: INC0000047, 3, 2, San Diego
---> Number: INC0000050, 1, 2, San Diego
---> Number: INC0000051, 1, 2, San Diego
---> Number: INC0000053, 1, 2, 937 Lomas Santa Fe Drive, San Diego, CA
         ****FS:CC Lab 1.5 encodedQuery refactored!-testEncodedQuery
[0:00:00.022] Total Time
```

Progress Report

1. Navigate to Lab Management > Report Lab Progress.

   Click I am done!

Lab Goal

Lab 1.6 Extending GlideRecord - Implementing Between

The purpose of this lab is to demonstrate:

- Extending an existing object is possible using JavaScript Prototypes
- ServiceNow objects can be extended using Prototypes
- Reusable objects and code function libraries can be created using this method
- Use of Fix Scripts to do unit testing

Problem Description

Many SQL WHERE functions exist with GlideRecord. For example:

```
SELECT * FROM <table> WHERE <field> = <value>
SELECT * FROM <table> WHERE <field> IN <valuelist>
SELECT * FROM <table> WHERE <field> LIKE <value%>
etc.
```

However, there is one function ostensibly missing from the list: The BETWEEN function:

```
SELECT * FROM <table>
WHERE <date field> BETWEEN ‘<date value 1>’ AND ‘<date value 2>’
```

For example:

```
SELECT * FROM incident
WHERE sys_created_on BETWEEN '2013-06-12' AND '2014-11-28'
```

This functionality can be currently written using the following method:

```
var <name> = new GlideRecord(<table>);
<name>.addQuery(<date field>, ‘>=’, ‘<date value 1>’);
<name>.addQuery(<date field>, ‘<=’, ‘<date value 2>’);
<name>.query();
```

It would be simpler and much easier to maintain if we had the following functionality:

```
var <name> = new GlideRecord(<table>);
<name>.addBetweenQuery(<date field>, ‘<date value 1>’, ‘<date value 2>’);
<name>.query();
```

The Solution – Extending GlideRecord With New Between Function

So let’s add the missing functionality!

1. Navigate to System Definition -> Script Include. The Script Include List View will be displayed.
2. Click on the New button. A new Script Include form will be displayed.
3. Fill in the following fields:
    a. **Name**: GlideRecordExtensionsCC
    b. **Accessable From**: All Application Scopes
    c. **Active**: checked
    d. Right-click on the form header to bring up the context menu and choose Save to save your work.

4. Now continue filling in the remaining fields:

    a. **Description**: addBetweenQuery - search for all records between two dates inclusive

    b. **Script**:

```
GlideRecord.prototype.addBetweenQuery = function(fieldToCheck, beginDate, endDate) {
    var beginDateCheck = gs.dateGenerate(beginDate); // convert to the appropriate string
    var endDateCheck = gs.dateGenerate(endDate); // ibid.

    this.addQuery(fieldToCheck, '>=', beginDateCheck);
    this.addQuery(fieldToCheck, '<=', endDateCheck);
};
```

    c. Click on the Update button to save your work.

Testing the .addBetweenQuery functionality

5. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.

6. Click on the New button. A new Fix Script form will be displayed.

7. Fill in the following fields:

    a. **Name**: GlideRecord Extension Between Test

    b. **Active**: checked

    c. **Run Once**: true

    d. **Description**: Test the .addBetween function for GlideRecord

e. **Script**:

```
gs.include('GlideRecordExtensionsCC');

var loggingSource = '\n\t****FS:CC Lab 1.6 addBetween-';

var startDate = new GlideDateTime('2013-06-12');
var endDate = new GlideDateTime('2014-11-28');

testBetween(startDate, endDate, loggingSource);

function testBetween(startDate, endDate, loggingSource) {

    // retrieve all active records between the two dates inclusive!
    var incidents = new GlideRecord('incident');
    incidents.addBetweenQuery('sys_created_on', startDate, endDate);
    incidents.addQuery('active', true);
    incidents.setLimit(2);  // lets only get a couple for the exercise
    incidents.query();

    gs.info('---> Records found: {0} {1}',
                incidents.getRowCount(),
                loggingSource + 'testBetween');

    while (incidents.next()) {
          gs.info('---> Records found: {0} - {1} {2}',
                      incidents.number,
                      incidents.sys_created_on,
                      loggingSource + 'testBetween');
    }
}
```

f. Right-click on the form header to bring up the context menu and choose Save to save your work.

8. Click on the Run Fix Script - CC button or Scroll down to the bottom of the form and click on the Run Fix Script related link to run the script; click the Ok button on the next form, and the Proceed button on the next form.

9. Your results should look something like the following:

```
*** Script: ---> Records found: 2

         ****FS:CC Lab 1.6 addBetween-testBetween
*** Script: ---> Records found: INC0000002 - 2013-06-15 22:30:06

         ****FS:CC Lab 1.6 addBetween-testBetween
*** Script: ---> Records found: INC0000003 - 2013-06-30 14:41:46

         ****FS:CC Lab 1.6 addBetween-testBetween
[0:00:00.026] Total Time
```

You could derive this same kind of functionality by using the following built-in functionality .addEncodedQuery. It would look like this:

```
incidents.addEncodedQuery("active=true^sys_created_onBETWEENjavascript:gs.dateGenerate('2013-06-
12','00:00:00')@javascript:gs.dateGenerate('2014-11-28','00:00:00')");
```

The encoded query would be difficult to read, and difficult to maintain. The .addBetweenQuery is much easier to read!

Progress Report

1. Navigate to Lab Management > Report Lab Progress.

   Click I am done!

Lab 1.7 Extending GlideRecord - Implementing .toJSON (Bonus)

Lab Goal

The purpose of this lab is to demonstrate:

- Building off of the .addBetween lab (Lab 1.5) add the ability for a GlideRecord to convert itself into a JSON object
- Add a .toObject method to convert a single GlideRecord record to an object array
- Add a .toObjectList method to convert a GlideRecord object into an array of object arrays
- The toObject, and toObjectList methods are needed by our JSON method.

Problem Description

The need arises from time-to-time to transmit the contents of an entire GlideRecord object via the web. This can be when:

- Using REST to transmit data to a Web Service external to ServiceNow
- Transmitting via Ajax from ServiceNow Server-side to the Browser
- Transmitting via events from a Business Rule to Server-side

And there are many more. JSON notation is the most efficient way of doing this.

So what is JSON? JSON stands for JavaScript Object Notation. According the www.json.org (the ECMA-404 definition site) it is a lightweight data-interchange format. It is language independent, but uses conventions familiar to most programmers.

The JSON JavaScript definition can be found at the following site:

http://www.ecma-international.org/publications/standards/Ecma-262.htm

In a nutshell JSON is an organized list of key-value pairs used to represent a data structure.

Here is a simple example of what JSON can look like:

```
{
   “type”:”Incident”,
   “number”:”101005”,
   “short_description”:”Car Demolished”,
   “description”:”The Loch Ness Monster does exist!  It came out of the water and
ate my car!”
}
```

As you can tell it has a very XMLish feel to it.

Adding the .toObject, .toObjectList, and .toJSON Functions

1. We will expand on the Script Include we created in Lab 1.6. Navigate to System Definition -> Script Include. The Script Include List View will be displayed.

2. Find the script GlideRecordExtensions Script Include that we created in our previous lab.

3. Modify the form to the following:

    a. **Description**:

    .addBetweenQuery - search for all records between two dates inclusive
    .toObject - Convert an GlideRecord record to an object
    .toObjectList - Convert an entire GlideRecord recordset to an object array
    .toJSON - convert the GlideRecod to a JSON object

b. Script. Add the .toObject, .toObjectList, and .toJSON functions:

```
GlideRecord.prototype.addBetweenQuery = function(fieldToCheck, beginDate, endDate) {

    var beginDateCheck = gs.dateGenerate(beginDate); // convert to the appropriate string
    var endDateCheck = gs.dateGenerate(endDate); // ibid.

    this.addQuery(fieldToCheck, '>=', beginDateCheck);
    this.addQuery(fieldToCheck, '<=', endDateCheck);
};

GlideRecord.prototype.toObject = function(recordToPackage) {
    var packageToSend = {};
    for (var property in recordToPackage) {
           try {
                 packageToSend[property] = recordToPackage[property].getDisplayValue();
           }
           catch(err){}  // eat any error and continue processing
    }

    return packageToSend;
};

GlideRecord.prototype.toObjectList = function() {
    var objectList = [];

    while(this.next()) {
           objectList.push(this.toObject(this));
    }

    // set it back to the beginning so that we can use if for other things
    this.restoreLocation();

    return objectList;
};

GlideRecord.prototype.toJSON = function() {
    return new global.JSON().encode(this.toObjectList());
};
```

4. Click on the Update button to save your work.

Testing the .toObject functionality

1. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
2. Click on the New button. A new Fix Script form will be displayed.
3. Fill out the form with the following:
    a. **Name**: CC Lab 1.7 toObject Test
    b. **Active**: checked
    c. **Run once**: checked
    d. **Description**: Fix Script demonstrate a method for converting a single GlideRecord record to an object array.
    e. **Script**:

```
gs.include('GlideRecordExtensionsCC');

var loggingSource = '---FS:CC Lab 1.7 toObject - ';

testConversion();

function testConversion() {
    var incidentRecords = new GlideRecord('incident');
    incidentRecords.addQuery('state', '!=', 7); // closed
    incidentRecords.setLimit(1); // We only want one record to play with
    incidentRecords.orderByDesc('number'); // get the most recent incident
    incidentRecords.query();

    if (incidentRecords.next()) {
          // convert the one recrod into an object array -
          // feed the current record to itself
          var incident = incidentRecords.toObject(incidentRecords);
          // you can reference the array elements either via array
          // nomenclature or by dot notation
          gs.info('---> {0} - {1} [{2}]',
                      incident['number'],
                      incident.short_description,
                      loggingSource + 'testConversion');



          // you can uncomment these lines to get a look at the
```

```
            // entire converted GlideRecord object (all must be uncommented)
            //gs.info('---> {0} [{1}]',
            //
     global.JSUtil.logObject(incidentRecords.toObject(incidentRecords)),
            //             loggingSource + 'test');
     }
}
```

   f. Right-click the top of the form to bring up the context menu and click Save to save your work.

4. Click on the Run Fix Script – CC button to run the script. The Run Fix Script pop-up will appear.
5. Click the OK button. The Run Fix Script Warning pop-up will appear. Click on the Proceed button. The results pop-up will appear.

You should see something like the following (your results may vary):

```
*** Script: ---> INC0010048 - This incident was created via webservice [---FS:CC Lab 1.7 toObject
 - testConversion]
[0:00:00.043] Total Time
```

The .toObject function will be used to individually convert each record of a full GlideRecord object into an array by the .toObjectList function.

Testing the .toObjectList functionality

1. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
2. Click on the New button. A new Fix Script form will be displayed.
3. Fill out the form with the following:
    a. **Name**: CC Lab 1.7 toObjectList Test
    b. **Active**: checked
    c. **Run once**: checked
    d. **Description**: Fix Script demonstrate a method for converting a full GlideRecord record to an object array list.
    e. **Script**:

```
gs.include('GlideRecordExtensionsCC'); // bring in our library

var loggingSource = '\n\t***FS:CC Lab 1.7 toObjectList-';

var startDate = new GlideDateTime('2013-06-12');
var endDate = new GlideDateTime('2014-11-28');

testConversion(startDate, endDate, loggingSource);

function testConversion(startDate, endDate, loggingSource) {

    // we will use our new .addBetweenQuery function as well!
    var incidents = new GlideRecord('incident');
    incidents.addBetweenQuery('sys_created_on', startDate, endDate);
    incidents.addQuery('active', true);
    incidents.setLimit(2);  // lets only get a couple for the exercise
    incidents.query();

    gs.info('---> Records found: {0} {1}',
                incidents.getRowCount(),
                loggingSource + 'testConversion');

    // create the object array list!
    var incidentList = incidents.toObjectList();

    gs.info('---> Records converted: {0} {1}',
                incidentList.length,
                loggingSource + 'testConversion');

    // let's see if we actually have the records!
    for (var i=0; i < incidentList.length; i++) {
          gs.info('---> {0} - {1} {2}',
                      incidentList[i].number,
                      incidentList[i].sys_created_on,
                      loggingSource + 'testConversion');
    }

    // You can go crazy and print off the whole thing.
    // But watch out as this could be really huge! :-)
    //gs.info('---> List: {0} {1}',
    //          global.JSUtil.logObject(incidentList),
    //          loggingSource + 'testConversion');
}
```

   f. Right-click the top of the form to bring up the context menu and click Save to save your work.

4. Click on the Run Fix Script – CC button to run the script. The Run Fix Script pop-up will appear.
5. Click the OK button. The Run Fix Script Warning pop-up will appear. Click on the Proceed button. The results pop-up will appear.

You should see something like the following (your results may vary):

```
*** Script: ---> Records found: 2

        ***FS:CC Lab 1.7 toObjectList-testConversion
*** Script: ---> Records converted: 2

        ***FS:CC Lab 1.7 toObjectList-testConversion
*** Script: ---> INC0000002 - 2013-06-15 17:30:06

        ***FS:CC Lab 1.7 toObjectList-testConversion
*** Script: ---> INC0000003 - 2013-06-30 09:41:46

        ***FS:CC Lab 1.7 toObjectList-testConversion
[0:00:00.071] Total Time
```

Testing the .toJSON functionality

Now finally we will test our JSON function!

1. Navigate to System Definition -> Fix Scripts. The Fix Scripts List View will be displayed.
2. Click on the New button. A new Fix Script form will be displayed.
3. Fill out the form with the following:
    a. **Name**: CC Lab 1.7 toJSON Test
    b. **Active**: checked
    c. **Run once**: checked
    d. **Description**: Fix Script demonstrate a method for converting a full GlideRecord record to a JSON object.
    e. **Script**:

```
gs.include('GlideRecordExtensionsCC'); // bring in our library

var loggingSource = '\n\t***FS:CC Lab 1.7 toJSON-';

var startDate = new GlideDateTime('2013-06-12');
var endDate = new GlideDateTime('2014-11-28');

testConversion(startDate, endDate, loggingSource);

function testConversion(startDate, endDate, loggingSource) {

    // we will use our new .addBetweenQuery function as well!
    var incidents = new GlideRecord('incident');
    incidents.addBetweenQuery('sys_created_on', startDate, endDate);
    incidents.addQuery('active', true);
    incidents.setLimit(2);  // lets only get a couple for the exercise
    incidents.query();

    // Now convert our recordset to JSON!
    gs.info('---> {0} {1}',
                incidents.toJSON(),
                loggingSource + 'testConversion');
}
```

   f. Right-click the top of the form to bring up the context menu and click Save to save your work.
4. Click on the Run Fix Script – CC button to run the script. The Run Fix Script pop-up will appear.
5. Click the OK button. The Run Fix Script Warning pop-up will appear. Click on the Proceed button. The results pop-up will appear.

You should see something like the following (your results may vary):

```
*** Script: ---> [{"active":"true","activity_due":"2016-04-24 21:57:26","additional_assignee_list
":"","approval":"","approval_history":"","approval_set":"","assigned_to":"Howard Johnson (howard.
johnson)","assignment_group":"Network","business_duration":"","business_service":"","business_stc
":"","calendar_duration":"","calendar_stc":"","caller_id":"","category":"Network","caused_by":"",
"child_incidents":"","close_code":"","close_notes":"","closed_at":"","closed_by":"","cmdb_ci":"Fi
leServerFloor2","comments":"2015-02-18 17:13:23 - System Administrator (Additional comments)\nAdd
ed an attachment\n\n","comments_and_work_notes":"2015-02-18 17:13:23 - System Administrator (Addi
tional comments)\nAdded an attachment\n\n","company":"","contact_type":"Email","contract":"","cor
relation_display":"","correlation_id":"","delivery_plan":"","delivery_task":"","description":"Use
r can't get to any of his files on the file server.","due_date":"","escalation":"Overdue","expect
ed_start":"","follow_up":"","group_list":"","impact":"1 - High","incident_state":"Awaiting Proble
m","knowledge":"false","location":"Salem OR","made_sla":"false","notify":"Do Not Notify","number"
:"INC0000002","opened_at":"2014-11-28 18:00:00","opened_by":"Joe Employee (employee)","order":"",
"parent":"","parent_incident":"","priority":"1 - Critical","problem_id":"PRB0000007","reassignmen
t_count":"1","rejection_goto":"","reopen_count":"","resolved_at":"","resolved_by":"","rfc":"","se
verity":"1 - High","short_description":"Unable to get to network file shares","skills":"","sla_du
e":"UNKNOWN","state":"Awaiting Problem","subcategory":"","sys_class_name":"Incident","sys_created
_by":"pat","sys_created_on":"2013-06-15 17:30:06","sys_domain":"global","sys_domain_path":"/","sy
s_id":"9d385017c611228701d22104cc95c371","sys_mod_count":"2,843","sys_tags":"","sys_updated_by":"
system","sys_updated_on":"2016-04-24 19:57:26","time_worked":"","u_ci_forensic":"","u_document_id
":"","u_requester":"System Administrator (admin)","u_source_table":"","u_thing1":"","u_thing2":""
,"upon_approval":"","upon_reject":"","urgency":"1 - High","user_input":"","variable_pool":"","var
iables":"","watch_list":"","wf_activity":"","work_end":"","work_notes":"","work_notes_list":"","w
ork_start":""},{"active":"true","activity_due":"2016-04-24 23:00:43","additional_assignee_list":"
","approval":"","approval_history":"","approval_set":"","assigned_to":"Beth Anglin (beth.anglin)"
,"assignment_group":"Network","business_duration":"","business_service":"","business_stc":"","cal
endar_duration":"","calendar_stc":"","caller_id":"Joe Employee (employee)","category":"Network","
caused_by":"","child_incidents":"","close_code":"","close_notes":"","closed_at":"","closed_by":""
,"cmdb_ci":"","comments":"","comments_and_work_notes":"","company":"ACME North America","contact_
type":"Email","contract":"","correlation_display":"","correlation_id":"","delivery_plan":"","deli
very_task":"","description":"I just moved from floor 2 to floor 3 and my laptop cannot connect to
 any wireless network.\n\t\t","due_date":"","escalation":"Moderate","expected_start":"","follow_u
p":"","group_list":"","impact":"1 - High","incident_state":"Active","knowledge":"false","location
":"Salt Lake City","made_sla":"false","notify":"Do Not Notify","number":"INC0000003","opened_at":
"2014-11-26 17:07:30","opened_by":"ITIL User (itil)","order":"","parent":"","parent_incident":"",
"priority":"1 - Critical","problem_id":"","reassignment_count":"2","rejection_goto":"","reopen_co
unt":"","resolved_at":"","resolved_by":"","rfc":"","severity":"1 - High","short_description":"Wir
eless widget access is down in my area","skills":"","sla_due":"UNKNOWN","state":"Active","subcate
gory":"","sys_class_name":"Incident","sys_created_by":"admin","sys_created_on":"2013-06-30 09:41:
46","sys_domain":"global","sys_domain_path":"/","sys_id":"e8caedcbc0a80164017df472f39eaed1","sys_
mod_count":"2,296","sys_tags":"","sys_updated_by":"system","sys_updated_on":"2016-04-24 21:00:48"
,"time_worked":"","u_ci_forensic":"","u_document_id":"","u_requester":"Joe Employee (employee)","
u_source_table":"","u_thing1":"","u_thing2":"","upon_approval":"","upon_reject":"","urgency":"1 -
 High","user_input":"","variable_pool":"","variables":"","watch_list":"","wf_activity":"","work_e
nd":"","work_notes":"","work_notes_list":"","work_start":""}]

         ***FS:CC Lab 1.7 toJSON-testConversion
[0:00:00.078] Total Time
```

Well that looks a bit messy! You can copy out all of the information between the square brackets and paste it into something like **Notepad++** with **XML Tools** plugin, and after formatting you can see what it really looks like!

```
{
     "active" : "true",
     "activity_due" : "2016-04-24 21:57:26",
     "additional_assignee_list" : "",
     "approval" : "",
     "approval_history" : "",
     "approval_set" : "",
     "assigned_to" : "Howard Johnson (howard.johnson)",
     "assignment_group" : "Network",
     "business_duration" : "",
     "business_service" : "",
     "business_stc" : "",
     "calendar_duration" : "",
     "calendar_stc" : "",
     "caller_id" : "",
     "category" : "Network",
     "caused_by" : "",
     "child_incidents" : "",
     "close_code" : "",
     "close_notes" : "",
     "closed_at" : "",
     "closed_by" : "",
     "cmdb_ci" : "FileServerFloor2",
     "comments" : "2015-02-18 17:13:23 - System Administrator (Additional comments)\nAdded an
attachment\n\n",
     "comments_and_work_notes" : "2015-02-18 17:13:23 - System Administrator (Additional comments)\nAdded
an attachment\n\n",
     "company" : "",
     "contact_type" : "Email",
     "contract" : "",
     "correlation_display" : "",
     "correlation_id" : "",
     "delivery_plan" : "",
     "delivery_task" : "",
     "description" : "User can't get to any of his files on the file server.",
     "due_date" : "",
     "escalation" : "Overdue",
     "expected_start" : "",
     "follow_up" : "",
     "group_list" : "",
     "impact" : "1 - High",
     "incident_state" : "Awaiting Problem",
     "knowledge" : "false",
     "location" : "Salem OR",
     "made_sla" : "false",
     "notify" : "Do Not Notify",
     "number" : "INC0000002",
     "opened_at" : "2014-11-28 18:00:00",
     "opened_by" : "Joe Employee (employee)",
     "order" : "",
     "parent" : "",
        "parent_incident" : "",
        "priority" : "1 - Critical",
        "problem_id" : "PRB0000007",
        "reassignment_count" : "1",
        "rejection_goto" : "",
        "reopen_count" : "",
        "resolved_at" : "",
        "resolved_by" : "",
        "rfc" : "",
        "severity" : "1 - High",
        "short_description" : "Unable to get to network file shares",
        "skills" : "",
        "sla_due" : "UNKNOWN",
        "state" : "Awaiting Problem",
        "subcategory" : "",
        "sys_class_name" : "Incident",
        "sys_created_by" : "pat",
        "sys_created_on" : "2013-06-15 17:30:06",
        "sys_domain" : "global",
        "sys_domain_path" : "/",
        "sys_id" : "9d385017c611228701d22104cc95c371",
        "sys_mod_count" : "2,843",
        "sys_tags" : "",
        "sys_updated_by" : "system",
        "sys_updated_on" : "2016-04-24 19:57:26",
        "time_worked" : "",
        "u_ci_forensic" : "",
        "u_document_id" : "",
        "u_requester" : "System Administrator (admin)",
        "u_source_table" : "",
        "u_thing1" : "",
        "u_thing2" : "",
        "upon_approval" : "",
        "upon_reject" : "",
        "urgency" : "1 - High",
        "user_input" : "",
        "variable_pool" : "",
        "variables" : "",
        "watch_list" : "",
        "wf_activity" : "",
        "work_end" : "",
        "work_notes" : "",
        "work_notes_list" : "",
        "work_start" : ""
},
{
        "active" : "true",
        "activity_due" : "2016-04-24 23:00:43",
        "additional_assignee_list" : "",
        "approval" : "",
        "approval_history" : "",
        "approval_set" : "",
        "assigned_to" : "Beth Anglin (beth.anglin)",
        "assignment_group" : "Network",
        "business_duration" : "",
        "business_service" : "",
        "business_stc" : "",
        "calendar_duration" : "",
        "calendar_stc" : "",
        "caller_id" : "Joe Employee (employee)",
        "category" : "Network",
        "caused_by" : "",
        "child_incidents" : "",
        "close_code" : "",
        "close_notes" : "",
        "closed_at" : "",
        "closed_by" : "",
        "cmdb_ci" : "",
        "comments" : "",
        "comments_and_work_notes" : "",
        "company" : "ACME North America",
        "contact_type" : "Email",
     "contract" : "",
     "correlation_display" : "",
     "correlation_id" : "",
     "delivery_plan" : "",
     "delivery_task" : "",
     "description" : "I just moved from floor 2 to floor 3 and my laptop cannot connect to any wireless
network.\n\t\t",
     "due_date" : "",
     "escalation" : "Moderate",
     "expected_start" : "",
     "follow_up" : "",
     "group_list" : "",
     "impact" : "1 - High",
     "incident_state" : "Active",
     "knowledge" : "false",
     "location" : "Salt Lake City",
     "made_sla" : "false",
     "notify" : "Do Not Notify",
     "number" : "INC0000003",
     "opened_at" : "2014-11-26 17:07:30",
     "opened_by" : "ITIL User (itil)",
     "order" : "",
     "parent" : "",
     "parent_incident" : "",
     "priority" : "1 - Critical",
     "problem_id" : "",
     "reassignment_count" : "2",
     "rejection_goto" : "",
     "reopen_count" : "",
     "resolved_at" : "",
     "resolved_by" : "",
     "rfc" : "",
     "severity" : "1 - High",
     "short_description" : "Wireless widget access is down in my area",
     "skills" : "",
     "sla_due" : "UNKNOWN",
     "state" : "Active",
     "subcategory" : "",
     "sys_class_name" : "Incident",
     "sys_created_by" : "admin",
     "sys_created_on" : "2013-06-30 09:41:46",
     "sys_domain" : "global",
     "sys_domain_path" : "/",
     "sys_id" : "e8caedcbc0a80164017df472f39eaed1",
     "sys_mod_count" : "2,296",
     "sys_tags" : "",
     "sys_updated_by" : "system",
     "sys_updated_on" : "2016-04-24 21:00:48",
     "time_worked" : "",
     "u_ci_forensic" : "",
     "u_document_id" : "",
     "u_requester" : "Joe Employee (employee)",
     "u_source_table" : "",
     "u_thing1" : "",
     "u_thing2" : "",
     "upon_approval" : "",
     "upon_reject" : "",
     "urgency" : "1 - High",
     "user_input" : "",
     "variable_pool" : "",
     "variables" : "",
     "watch_list" : "",
     "wf_activity" : "",
     "work_end" : "",
     "work_notes" : "",
     "work_notes_list" : "",
     "work_start" : ""
}
```

Cool huh?!?! ☺

Progress Report

Navigate to Lab Management > Report Lab Progress.

Click I am done!