Introduction

This documentation presents **30** requirements and solutions that i have already published, it differs only in the way the information is presented, each requirement with a script is illustrated with a **diagram** that you can **visualize** to better understand the code.

A page is divided into three vertical sections, the first one contains the script, the second section contains commentary on each line of the script, and the last section is where you can find the diagram.

Topics covered in this documentation are for servicenow beginners, you will learn the following : **clients scripts, ui macro, notifiaction, events, business rules, scheduled script, ui actions , ui policy and integration.**

For more information about the requirement read the content table and consult the next page to know **How to read this documentation.**

Table of Contents

How to Read this documentation

1 Javascript code

```
1.1 const colors = ['red', 'green', 'blue', 'yellow', 'pink',
1.2 'purple'];
const body = document.querySelector('body');
body.style.backgroundColor = 'violet';
const button = document.querySelector('button');

button.addEventListener('click', changeBackground);

function changeBackground(){
const colorIndex= parseInt(Math.random()*colors.length);
body.style.backgroundColor = colors[colorIndex];
}
```

This first section contains a Javascript Code, this code here it is changing the body background color based on the user action. when the user click on the button, it will change the background color.

1.1 , 1.2, 1.3 are not line numbers, they should be seen as steps in our code.

For each step you have the comments in the the second section,of the page, details explanation of what the code is exactly doing..

The third section contains the most interesting part which is the diagram of the script, refer to the step number to see the code behaviour in the diagram.

for example step 1.1 we create an array which contains some data, the dot for the step will only come afterwards.

Green arrow indicates in which direction the code is executed, for example if you are assigning a value to variable, the arrow will point in the direction of the variable where you will actually store the value.

Thin black line indicates how one element it is reused multiple times in the code. for example.the **array** object is used first at step 1.1 when we are creating it, then at step 1.7 when we look for the **arrays length** and third time at step 1.8 when we actually use it **to assign a random color** to the body background element.

Properties of a class are *italicized* and api's are in a black boxes.

Follow these guidelines for the rest of tthe documentation, some exceptions are made based on the requirement. for example servicenow to servicenow integration do not have any script.

1.1 We begin by creating **colors Array** which contains the following colors : red, green, blue, yellow, pink and purple. This array to pick any color assign it to the body background

1.2 Then create a **body objec**t, for that use **dom api** and its method such **querySelector(),**

At step **1.3** assign **violet** as a initial body background color, note we use **style.background property.**

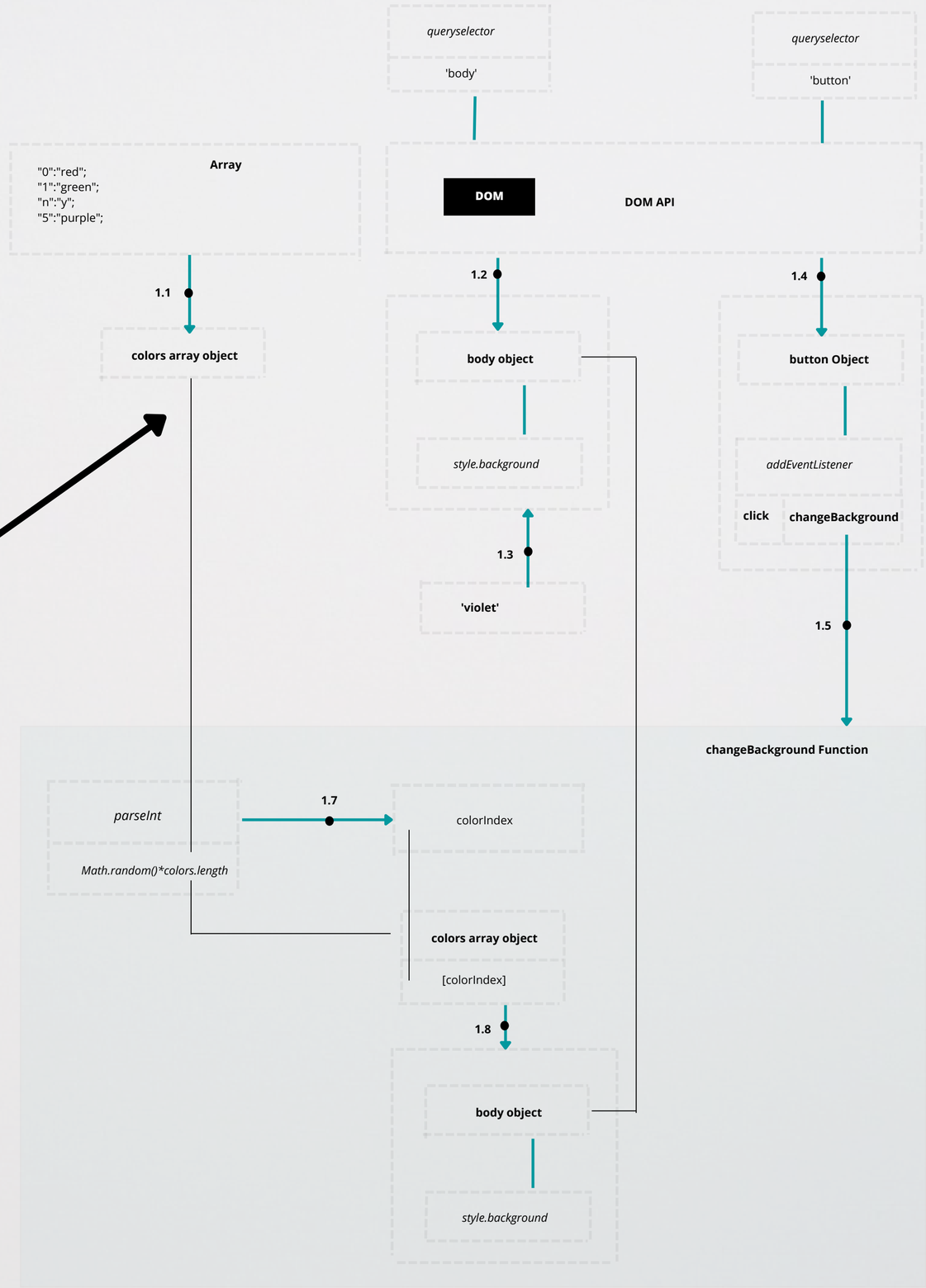
1.4. create **button object**, the same way we have created the body object, use dom api and querySelector method in order to create it.

1.5 use **addEventListener** method to listen to the **click** event and call a changeBackground **callback function** on a click.

Callback Function :

1.7 use parseInt method to round the random colors index number, the highest possible value is 5. as the length of the array is 5

1.8 use the colorsIndex as an **index** for the **colors Array** and assign array value to the background colors, the background color is determined based on the the random number generated and will change based on the user clicks



01

When the Caller is VIP set the incident urgency to High and alert a message.

Caller : not vip
Urgency : medium

1 User or system changes the caller field value, and the caller is VIP

Client script is written on the **caller field**

2

```
function onChange(control, oldValue, newValue, isLoading,
isTemplate)
{

if (isLoading || newValue === '') { return; }

var user = g_form.getReference('caller_id', vipFunction);


function vipFunction(user){

if(user.vip=='true'){

g_form.setValue('urgency',1);
g_form.addInfoMessage
('the caller of this incident is vip, set the priority to High.');
                                    }

                                                    }
}
```

Caller : **vip**
Priority : **High**
3 Control is given to the user.

2.3 The first block will return an **user object** which we can dotwalk to get values from the reference field, in our code we are sending the caller as input for the reference field. and aslo call a callback function.
Use GlideForm API and getReference method to get referencen field data, here it will the caller field, caller field is sent as parameter in the getReference method, the second parameter is the vip callback function which can use the user object to manipulate values.

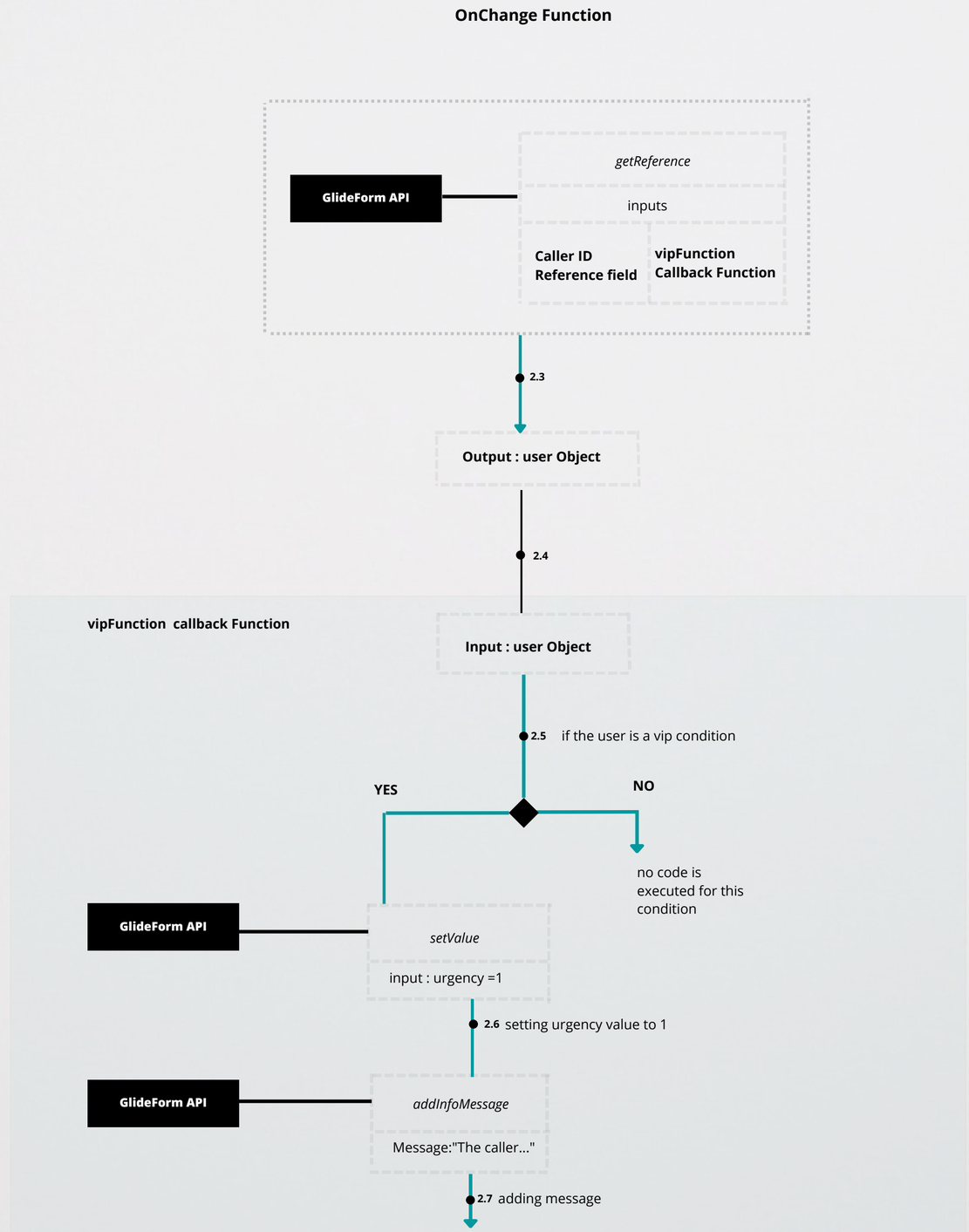
Vip Callback function

2.4. Sending user **Object** as input in the callback function

2.5. The user **object** is dotwalkable, we can check the vip field is activated or not, use the if condition to check user.vip is true if yes then we execute the code further.

2.6. Setting the **urgency** field value to 1.

2.7. Adding a info message, the caller of this incident is vip and set the priority to high. Changing the value of urgency will automatically change the priority field value.



02

ServiceNow to ServiceNow Integration

Even though there is a data repilcation application in servicenow dedicated to integration inbetween servicenow intances, here is a another way to integrate two servicenow instances.

There isn't only one particular grant flow to perfom integration in servicenow, in this exercise, we will see how we can use OAuth Code Grant flow to do the integration.

Procedure:

The explanation below explains how to integrate servicenow instance A and Instance B, instance a will have the following url : https://instance_a_name/ and b https://instance_b_name/

Section A represents **servicenow OAuth Provider Application**, that's the application which will provide access to **servicenow API'S**, once you have configured the OAuth Provider, you will get the **credentials** which will be further used in the grant flow for communication. Look at the **step 1**, to see how it is being used, an authorization request is sent from Instance A to B with the credentials. "Client id and Client Secret"

Begin first by creating, servicenow OAuth Provider, in Instance B navigate to OAuth, under Application registry and create **OAuth API for external clients** and note down the Client ID and Client Secret, Also insert the redirect URL : https://instance_a_name/oauth_redirect_do

The redirect url is used in order to send a Code from Instance B to A, see **step 2 and 3**, a user approves instance A to access instance B, after credentials verification, instance B sends a code to the redirection url.

Section B represents **servicenow OAuth 2.0 Application,** and here you provide the credentials provided OAuth Provider App from instance B, also the **authoriszation url** and **token url.**

https://instance_b_name/oauth_auth.do for Authorization URL.
https://instance_b_name/oauth_token.do for Token URL.

Authorization url is used to send authorize request in the **step 1** and Token url is used to exchange a code for a token, **step 4.**
In the instance A Navigate to OAuth and under the application registry and create an OAuth Application (click on Connect to theThird party OAuth Provider), by default when you create an OAuth 2.0 client application, it will aslo create a OAuth entity profile section **B.1** Which contains the **application profile** information and as well the **oauth profile scopes (Section B.2)**

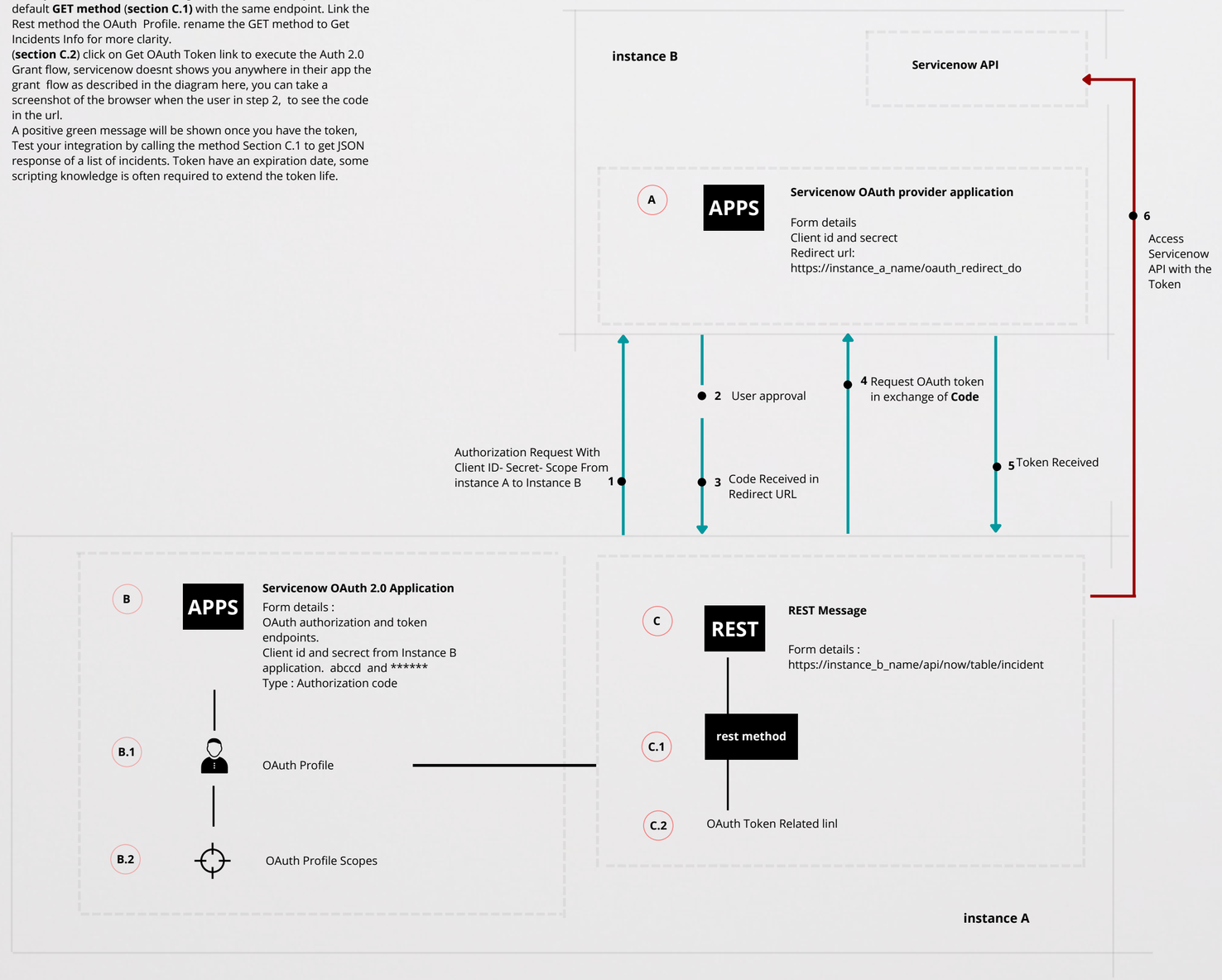
Scopes tells you what kind of operation you can do with the provider application, for example create, read records, in this exercise there will be no scopes added but if you integrate servicenow with a third party application such as Salesforce or Linkedin, detais of scopes will be provided by the service provider.

Section C, Navigate to outbound REST message and create A **REST message.** To know about what is REST, see the previous page. REST has several methods, like GET, PUT, POST and DELETE, this is not be confused with the profile scopes, for example you can send a request to GET user profile information but you may requird read profile info scope to actually receive profile information.

In this exercise, we will use the following endpoint or url for the REST Message Endpoint :

https://instance_b_name/api/now/table/incident

when you create the REST Message , it will automaitcally create the default **GET method** (**section C.1)** with the same endpoint. Link the Rest method the OAuth Profile. rename the GET method to Get Incidents Info for more clarity.
(**section C.2**) click on Get OAuth Token link to execute the Auth 2.0 Grant flow, servicenow doesnt shows you anywhere in their app the grant flow as described in the diagram here, you can take a screenshot of the browser when the user in step 2, to see the code in the url.
A positive green message will be shown once you have the token, Test your integration by calling the method Section C.1 to get JSON response of a list of incidents. Token have an expiration date, some scripting knowledge is often required to extend the token life.



03

If the current incident is a child incident, add a info message with a clickable parent incident number

1 User clicks on a incident record
Parent Incident : INC0125148

2 Data loads from the database

3 Client script execution and loading data from the database.

```
function onLoad()
{
3.1 var parent = g_form.getReference('parent_incident',parentFunc);


    function parentFunc(parent){
3.2 var p = g_form.getValue('parent_incident');
3.3 if (p!=""){
3.4 g_form.addInfoMessage('This is a child incident of '+'<a
3.5 href=\'https://dev64478.service-now.com/nav_to.do?
    uri=incident.do?sys_id='+parent.sys_id+'\'
    target=\'_blank\'>'+parent.number +'</a>');

                }
                                                    }
}
```

4 Message : This is a child incident of **INC0125148**
Control is given to the user.

.3 The first block will return an parent object which we can dotwalk to get values from the reference field, in our code we are sending the parent_incident field as input for the reference field. and also call a callback function.
Use GlideForm API and getReference method to get referencen field data, here it will the parent incident field, this fielld is sent as parameter in the getReference method, the second parameter is the **parentFunct** callback function which can use the user object to manipulate values.

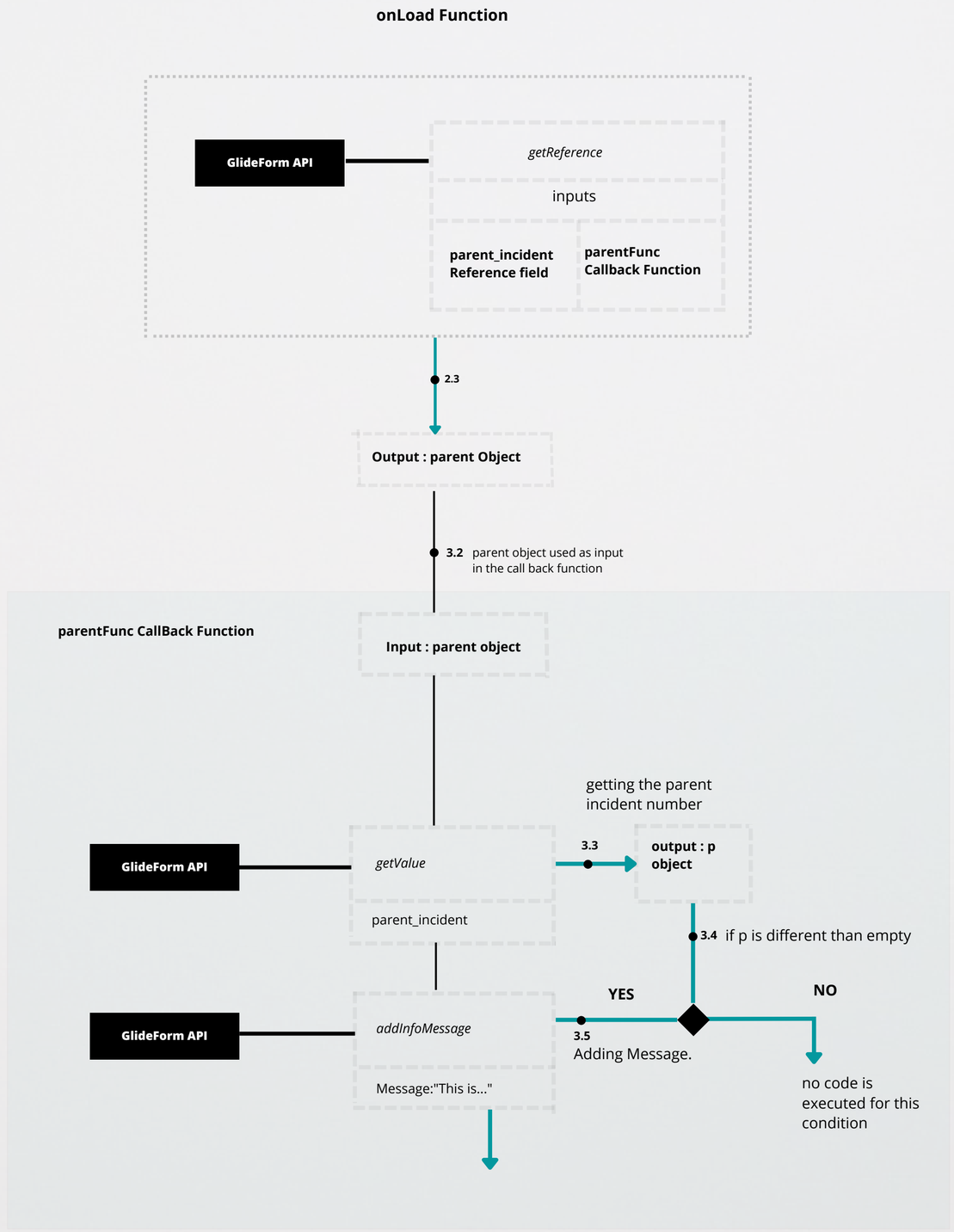
ParentFunct Callback function

3.2. Sending parent **Object** as input in the callback function

3.3 Getting the parent field value, the incident number and storing **p variable.**

3.4. Checking if **p** is not empty, if yes then executing the next code

3.5. Adding a message which contains the **parent incident number** , **sys id** , we use the **parent object** to get the sys_id and number like **parent.sys_id** and **parent.number**



04

If the incident short description starts with the string 'Database', set database as category

Short Description: Email server is down
Category : Network

User or system changes the short description to -> Database, email server is down

1

Client script execution
Client script is written on the **short description field**

2

```
function onChange(control, oldValue, newValue,
isLoading, isTemplate)
{

if (isLoading || newValue === '') {return;}

2.3 var short_desc= g_form.getValue('short_description');
2.4 var position = short_desc.search('Database');


2.5 if (position==0){

2.6 g_form.setValue('category','database');

                                }

}
```

3

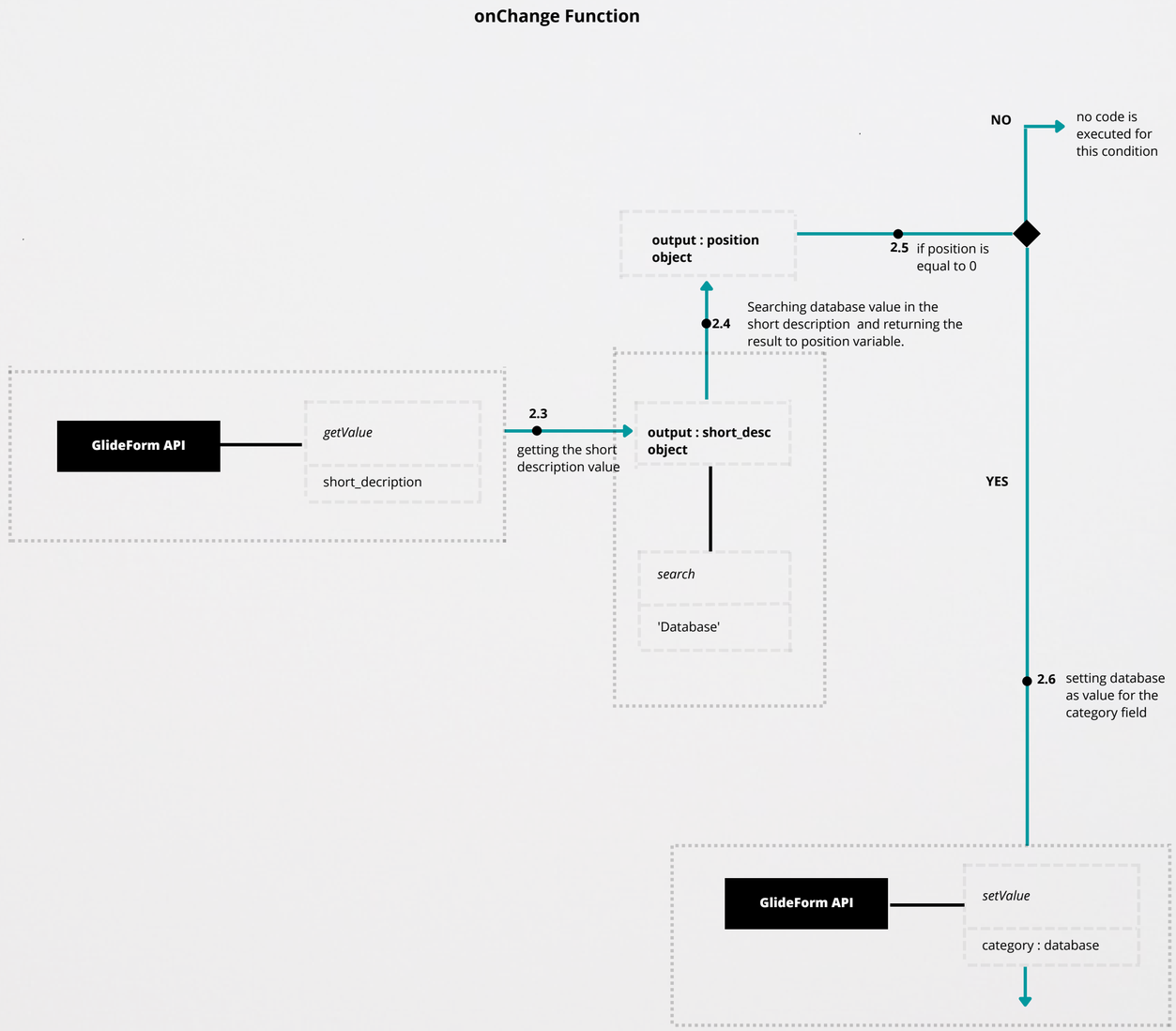
Short Description : Database, email server is down
Category : **Database**
Control is given to the user.

2.3 Use GlideForm API and its method getValue to get the short_description value, and store the value in the short description variable

2.4. Use **Search Method,** to check if the value database is available in the short description, store the return value in the **position variable**

2.5 if the **position i**s equal to **0**, it means, the database value is available at the **first position** in the **short description variable.**

2.6 Setting **database** as value for the **category field.**



05

onclick of a checkbox, open the list of active users in a new window

onChange Function

1 User checks a custom check box

Client script is written on the **check box field and for incident table**

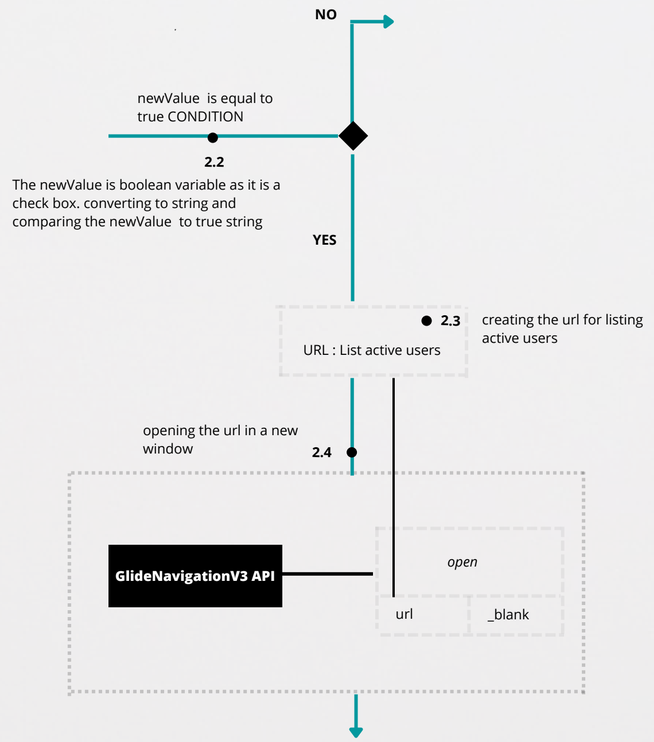
2
```
function onChange(control, oldValue, newValue,
isLoading, isTemplate) {

if (isLoading || newValue === '') {return;}

2.2 if (newValue.toString() == 'true') {

2.3 var url = "sys_user_list.do?sysparm_query=active=true";
2.4 g_navigation.open(url, "_blank");


}
}
```

3 Opens a new window with the list of active users

2.2 We begin by checking if the checkbox is checked, if is checked it will be true otherwise false, but it is boolean value, we need to convert it to string and check with the true literal string.if the condition is true, we execute the code further

2.3 creating an url for listing a list of active users

2.4 use GlideNavigation API and its method open in orde to open an URL



06

Remove any attached files that are more than 15kb in incident and service catalog request tables

1 User creates a record.

2 Database is updated.

3 After Business Rule Execution

```
3.1 var grsys = new GlideRecord("sys_attachment");
3.2 grsys.addQuery("table_name", "IN"
    ,"incident,sc_request");
3.3 grsys.query();

3.4 while(grsys.next()) {
3.5 if(parseInt(grsys.size_bytes) >15000)
3.6 grsys.deleteRecord();

    }
```

4 All attachements with the size more than 15kb are deleted from the incident and service request table.

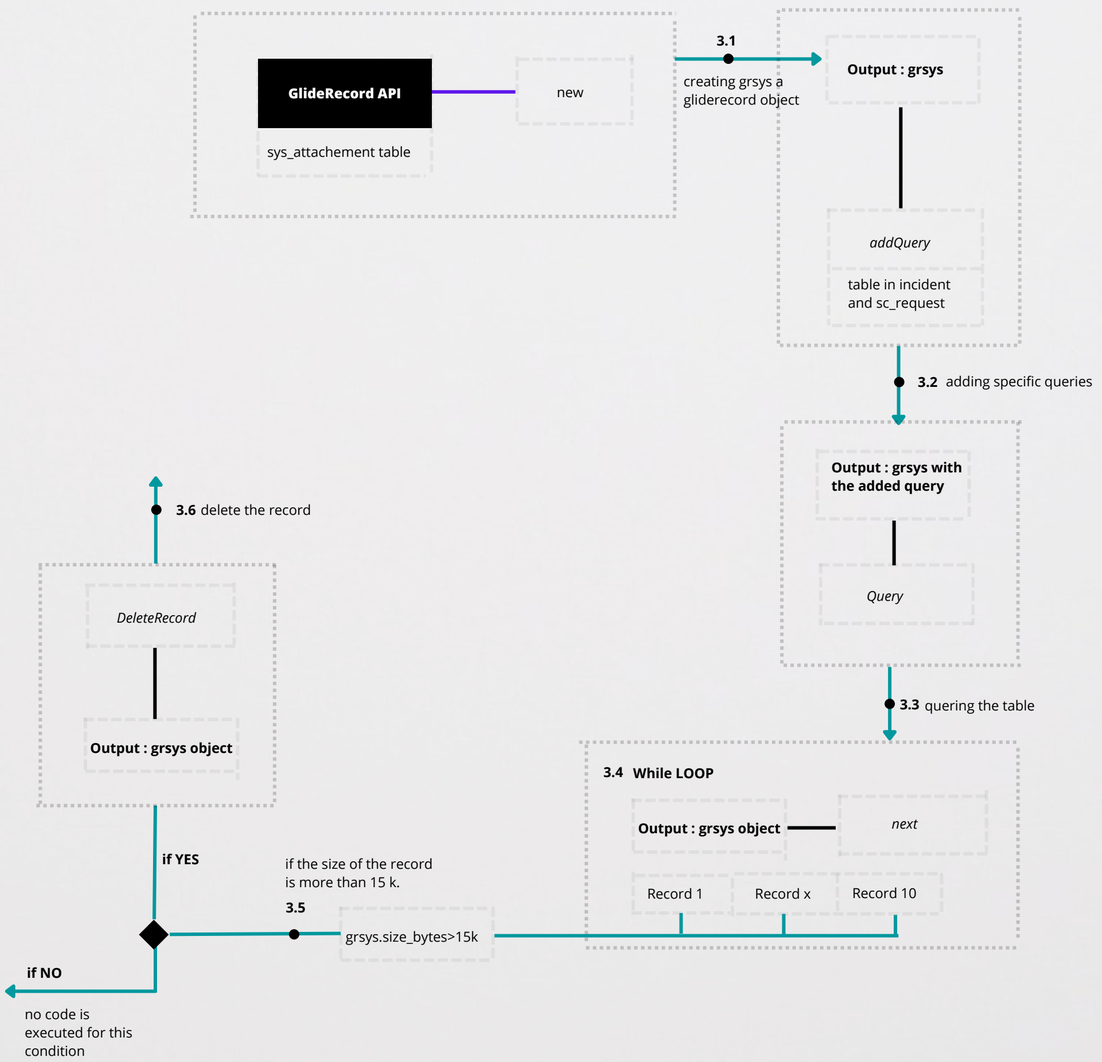
3.1 Creating a **GlideRecord Object** of sys_attachement table and storing the value in **grSys** variable.

3.2 Adding a **query, lookups** only records where table name is **incident** and **sc_request**

3.3 quering with **grSys** Object

3.4 for each record found in the query check if the byte size is superiror than 15kb

3.5 if yes, then delete the record with deleteRecord method.



07

Show an error message if the incident priority is critical

1 User opens an incident record
Priority : **critical**

2 Loading data from the database.

3 The client script is written for the **incident** table.

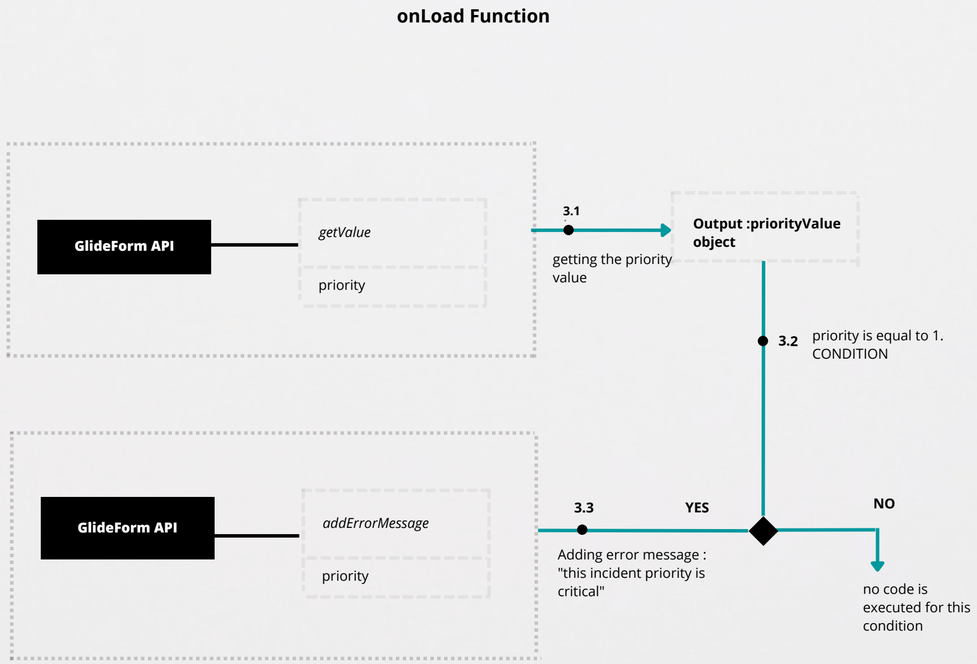
```
function onLoad(){
3.1 var priorityValue = g_form.getValue('priority');

3.2 if(priorityValue=='1'){
3.3 g_form.addErrorMessage('this incident priority is
critical');
                                    }
}
```

4 Message : this incident priority is critical'
Control is given to the user.

3.1. Use GlideForm API and its method getValue for getting the priority field value and assign the value priorityValue field .

3.2 create the if Condition to check if the priority is critical

3.3. If the priority Value is 1 , it means it is critical, add an error messge : "This incident priority is critical"



08

For every hour automatically assign analytics_admin role for users who don't have analytics_admin or analytics_viewer role.

The goal is to assign a role to specific users within a time period, write the script which will check the conditions, make sure that these users don't have analytics_admin or analytics_viewer role, use the the script in a scheduled script and schedule it for every hour.

Code explanation Part 1

Scheduled script
for every hour

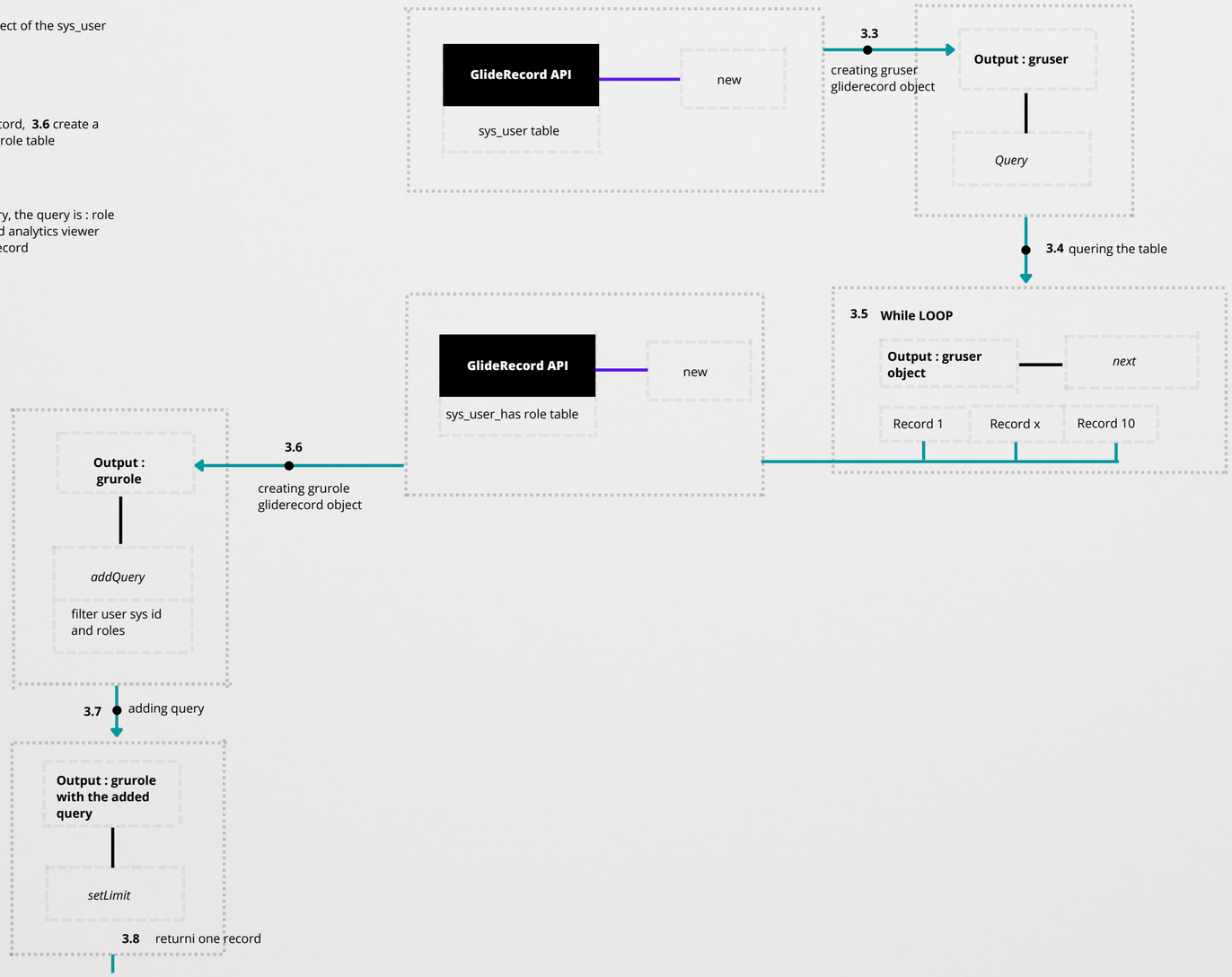
```
giveRole();

   function giveRole(){

   try{
3.3   var gruser = new GlideRecord("sys_user");
3.4   gruser.query();



3.5   while(gruser.next()) {

3.6   var grurole = new GlideRecord("sys_user_has_role");
3.7   grurole.addQuery("user", gruser.sys_id);
      grurole.addQuery("role.name",
"IN","analytics_admin,analytics_viewer");
3.8   grurole.setLimit(1);
      grurole.query();



      if(!grurole.next()){
      grurole.initialize();
      grurole.user = gruser.sys_id;
      grurole.setDisplayValue("role", "analytics_admin");
      grurole.insert();
     }
    }
   }
   catch(ex){ gs.info(ex);}
}
```

3.3 create grUser a GlideRecord Object of the sys_user table.

3.4 query with the object

3.5 in a while loop, for each user record, **3.6** create a GlideRecord Objec of sys_user_has_role table

3.7 Filter the result by adding a query, the query is : role name should be analytics admin and analytics viewer set limit to one to return only one record



08

For every hour automatically assign analytics_admin role for users who don't have analytics_admin or analytics_viewer role.

The goal is to assign a role to specific users within a time period, write the script which will check the conditions, make sure that these users don't have analytics_admin or analytics_viewer role, use the the script in a scheduled script and schedule it for every hour.

Code explanation Part 2

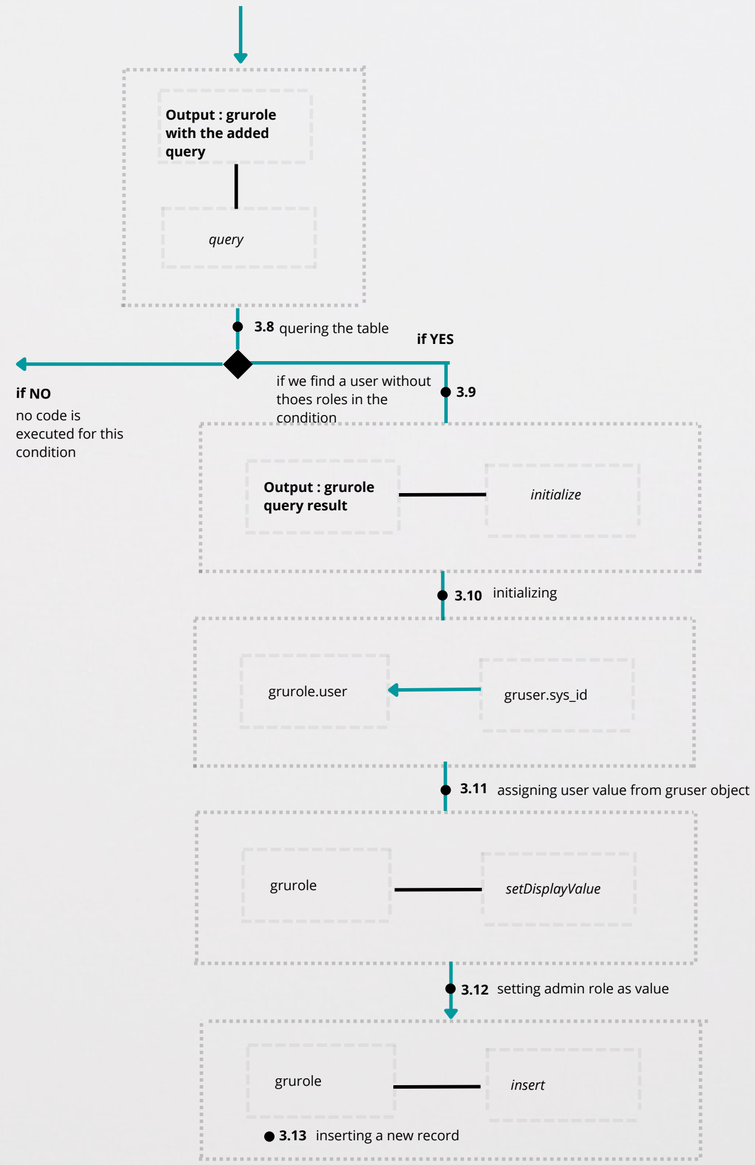
Scheduled script
for every hour

```
giveRole();

   function giveRole(){

   try{
      var gruser = new GlideRecord("sys_user");
      gruser.query();



      while(gruser.next()) {

      var grurole = new GlideRecord("sys_user_has_role");
      grurole.addQuery("user", gruser.sys_id);
      grurole.addQuery("role.name",
"IN","analytics_admin,analytics_viewer");
      grurole.setLimit(1);
3.8   grurole.query();



3.9   if(!grurole.next()){
3.10  grurole.initialize();
3.11  grurole.user = gruser.sys_id;
3.12  grurole.setDisplayValue("role", "analytics_admin");
3.13  grurole.insert();
       }
      }
     }
     catch(ex){ gs.info(ex);}
}
```

3.8 query to get the result

3.9 if there is a user without the role defined in the query, execute further the code

3.10 initialize grurole object, setvalue for the grurole user field with **3.11** grUser object sys_id and **3.12** setdispay value for role field the value analytics admin, and **3.13 i**nsert a record, we have added a new role to the user is sys_user_has_role table.



09

In a problem record, hide Notes section when problem state is equal to '102'.

The onchange client script will if there is a new value for state field, here if the database value is equal to 102, it wil hide the note changing section display value to false, this client script will also check the state value while the form is loading.

1 User opens a problem record

2 Data loads from the database

Client script execution
The client script is written for the **problem** table.

3

```
function onChange(control, oldValue, newValue, isLoading,
isTemplate) {
 if (isLoading) {
 if(g_form.getValue('state') == '102'){
 g_form.setSectionDisplay('notes', false);
 }
 return;
}

 if (newValue == "102") {
 g_form.setSectionDisplay('notes', false);
 } else {
 g_form.setSectionDisplay('notes', true);
 }
}
```

4 state is 102
notes section is hiddent

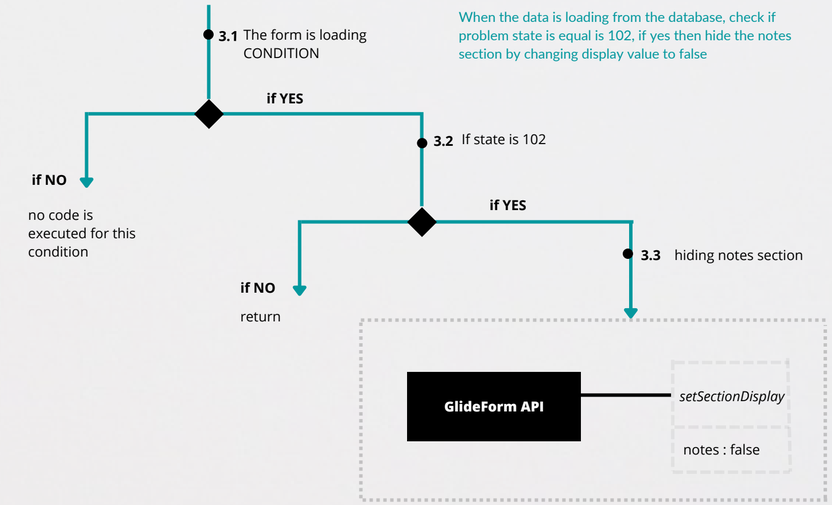
3.1 checking with a if condition is the form is loading or not if yes
1.2 use glideform api and getValue method in the second condition to get the state value and compare it to 102.

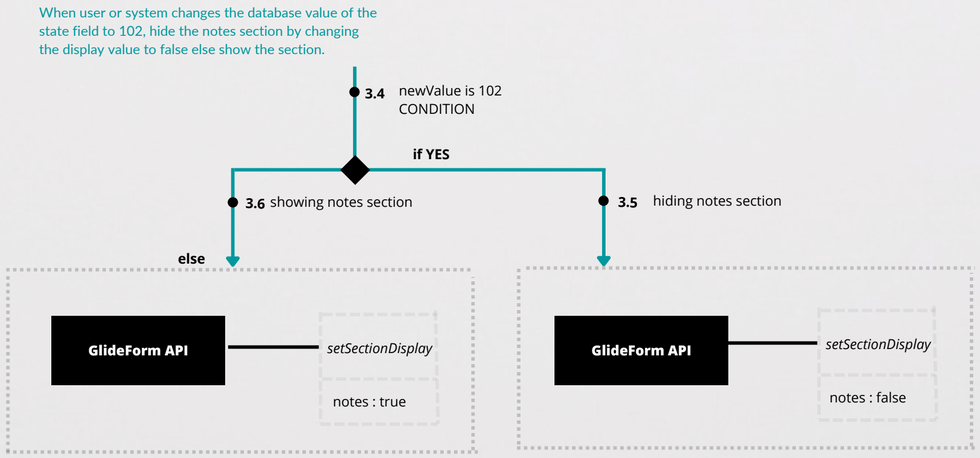
1.3 if the second condition is true, use glideform api and setsectionDisplay method to hide the section.

3.4 now we want to do the same logic describle previously when we there is a new value equal to 102 meaning where there is a change. if the contion newValue equal to 102 is true,

.**3.5** hide the section with glideform api and setSectionDisplay method.

3.6 else make it visiblle using the glideform api and setSectionDisplay value, the notes value will be false.





10

Calculate elapsed time from the Incident record creation and add the result to duration type field

Add a duration type field to the incident form, the field will contain the elapsed time and by default the database name for this field is u_duration

1 Database.

2 Display business rule

```
(function executeRule(current, previous /*null when async*/) {

  var start = new GlideDateTime(current.sys_created_on);
  var stop = new GlideDateTime();

  var duration = GlideDateTime.subtract(start,stop);
  current.u_duration=duration;

})(current, previous);
```

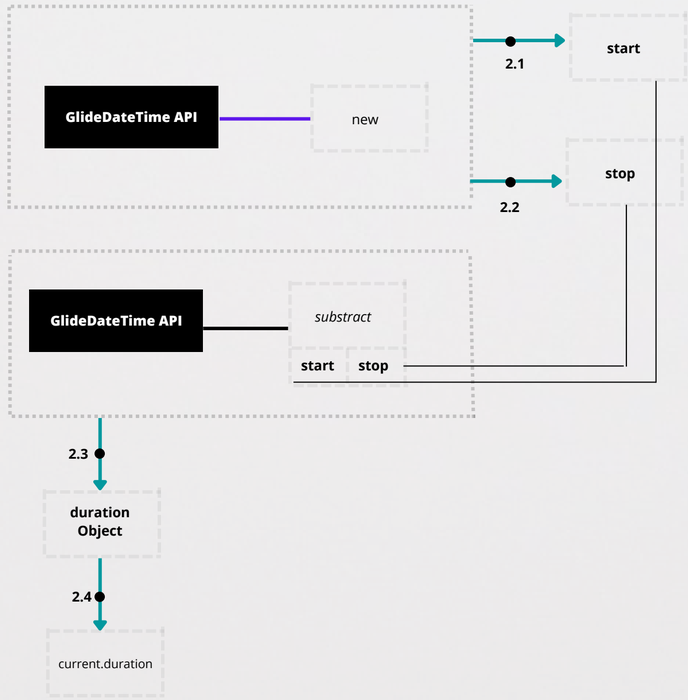
3 User creates a record.
duration : Days 75 Hours 00 09 49

2.1 Creating the start date GlideDateTime Object and storing the value in the start variable

2.2 Creating the stop date GlideDateTime Object and storing the value in the end variable

2.3 Use GlideDateTime API and it s substract method to substract start and stop, storing then the value in the duration field

2.4 Assigning the duration field value to current.duration field value



11

Extract IP address from the short description and assign the value to another field called Caller IP.

Incidents gets created with the following short description (sample):

[RESOLVED] CafeAddressCafeXYZ- ABCD - MNOP 96.233.79.10 is unavailable by SMTP

---

1 User updates an incident record.
short description : [RESOLVED] CafeAddressCafeXYZ- ABCD - MNOP 96.233.79.10 is unavailable by SMTP
Caller ip :

Before Business Rule Execution on the incident table

2

```
var r = /\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b/;


2.2 var a = current.getValue('short_description');
2.3 var t = a.match(r);
2.4 current.setValue('ip_address_field', t[0]);
```

2.1 create **r** variable which will contains a regex expressions

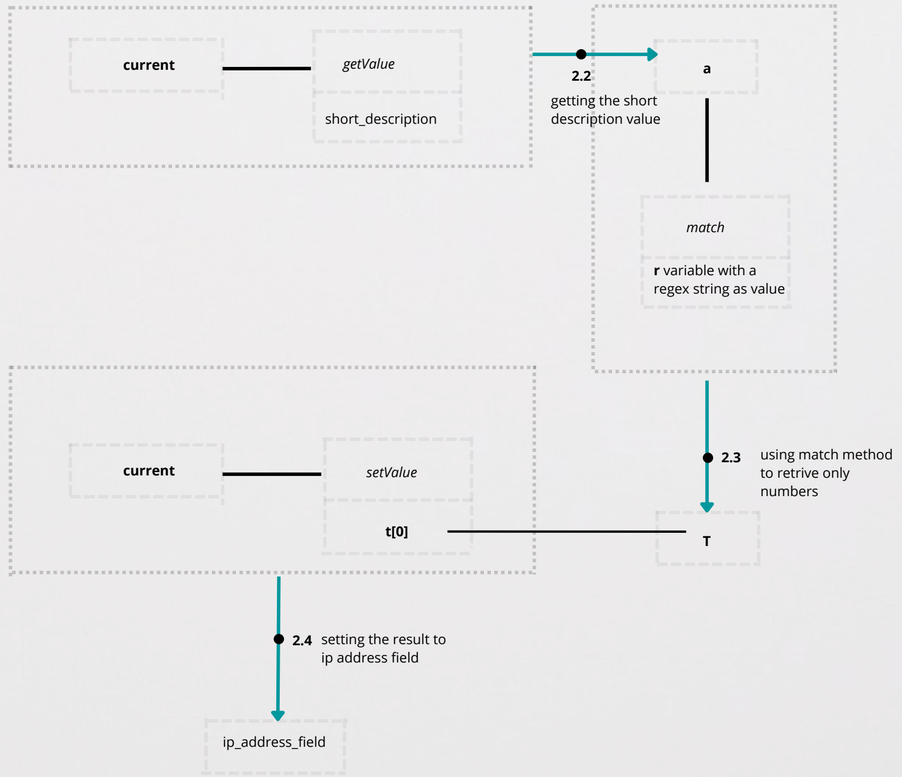
2.2 Get the short description field value, with the current object, current to the incident table.

3.2 use **match** method and give as inputs the **regex value** availabe in **r** variable, store the result in the t variable.

2.4 use current object again, to set value for the ip address field, the value in the index 0 of t array object.

3 Caller ip field value :
96.233.79.10

4 Database is updated.



12

Copy the entire journal from Change Request to a RITM when REQ Task is updated.

Code Part 1 : Explanation

1 User updates REQ task

2 Database is updated.

3 After Business Rule Execution

```
(function  executeRule(current,  previous  /*null
when async*/) {

3.1 var grcr = new GlideRecord('change_request');
3.2 grcr.addQuery('sys_id', current.parent);
3.3 grcr.query();


3.4 if(grcr.next()){
var notes = grcr.work_notes.getJournalEntry(-1);
var grsc = new GlideRecord('sc_req_item');
grsc.addQuery('request', current.sys_id);
grsc.query();

if (grsc.next()){
        grsc.work_notes = notes;
        grsc.update();
    }
  }

})(current, previous);
```

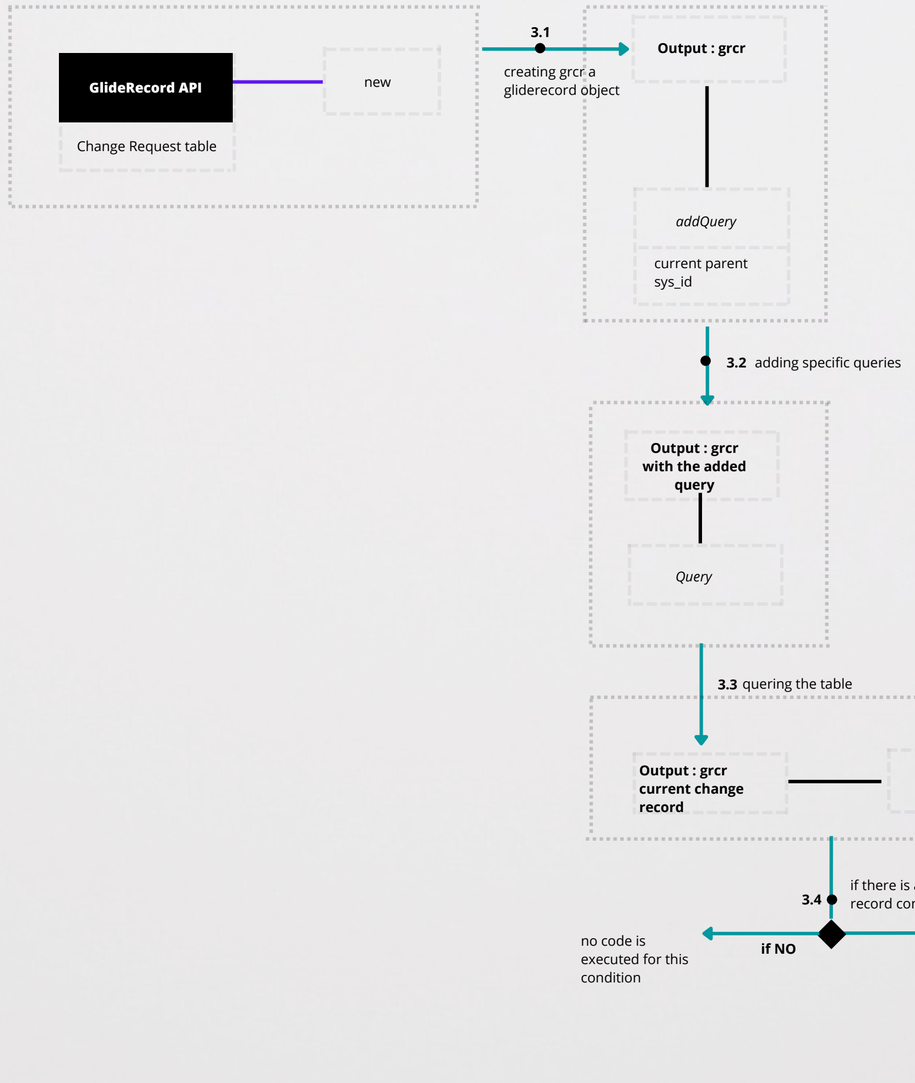
4 Worknotes is updated in the ritm record

3.1 create a grcr a **GlideRecord Object** of change request table

3.2 add query with the current parent field value, which is the change request record

3.3 query with **grCr** object

3.4 If there is a record with the query mentioned before execute the code further



12

Copy the entire journal from Change Request to a RITM when REQ Task is updated.

Code Part 2 : Explanation

1 User updates REQ task

2 Database is updated.

After Business Rule Execution

3

```
(function  executeRule(current,  previous  /*null
when async*/) {

var grcr = new GlideRecord('change_request');
grcr.addQuery('sys_id', current.parent);
grcr.query();


3.4 if(grcr.next()){
3.5 var notes = grcr.work_notes.getJournalEntry(-1);
3.6 var grsc = new GlideRecord('sc_req_item');
3.7 grsc.addQuery('request', current.sys_id);
grsc.query();

if (grsc.next()){
        grsc.work_notes = notes;
        grsc.update();
    }
  }

})(current, previous);
```

4 Worknotes is updated in the ritm record

3.4 If there is a record with the query mentioned before, execute the code further

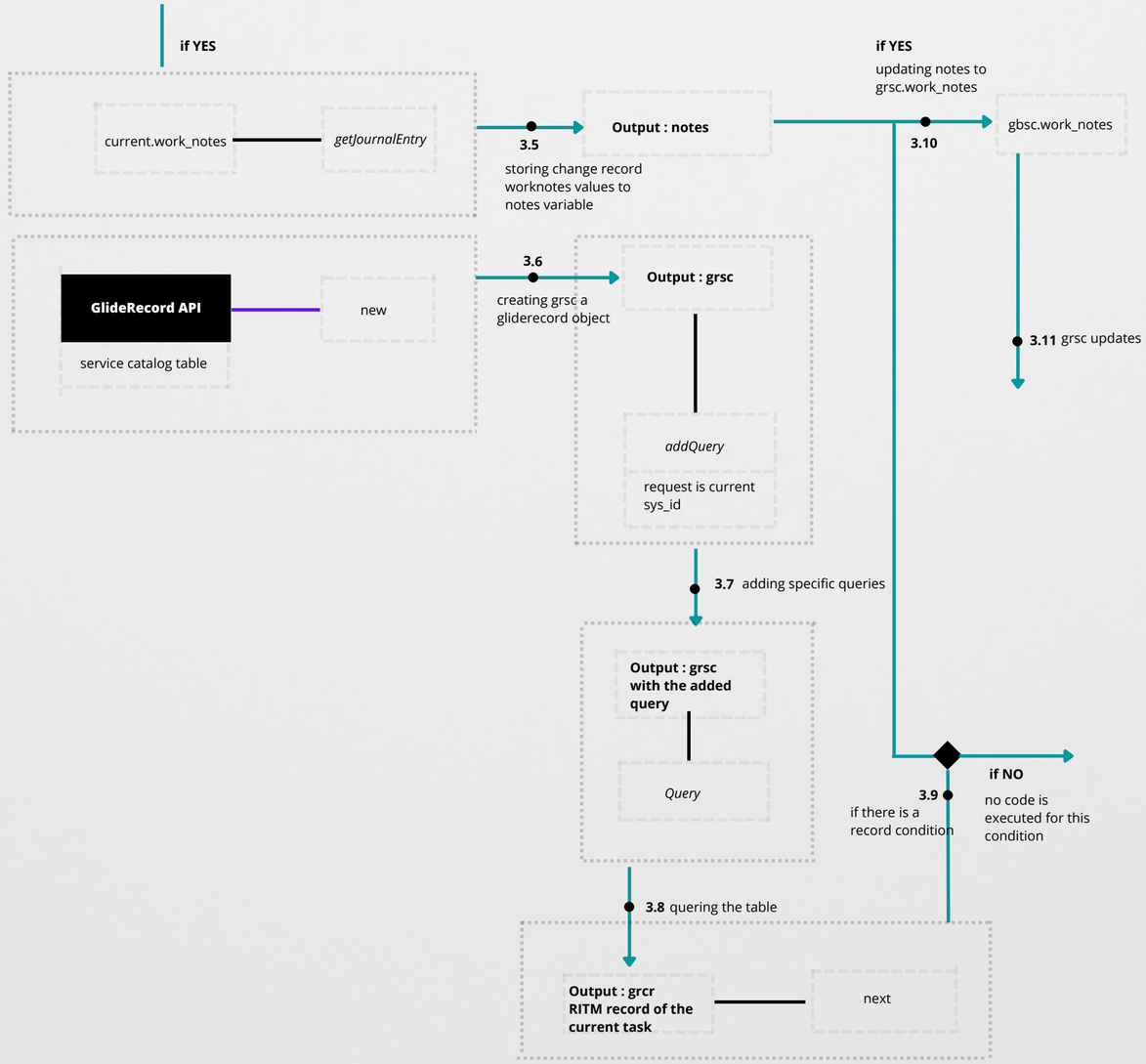
3.5 use **getJournalEntry Method** and **grcr object** to get the entire worknotes from the change request , store the value into the notes variable.

3.6 creating **grSc** a **GlideRecord Object** of **sc_req_item** table

3.7 add query witt the **current sys_id value** and **request field** for finding the related request **ritm**.

3.8 query the table

3.9 if there is a RITM, **3.10** assign **notes** value to **grsc.worknotes** which ritm **worknotes field** and **3.11** update



13

Create a child incident from the Incident Form

Create an ui action in the incident table, give it a name, choose type form button and select show insert and update.

create a child incident

Parent Incident

1 User clicks on the ui action

UI Action Script
The ui action script is written on the **incident table**

2

```
2.1 var grinc = new GlideRecord('incident');
2.2 grinc.short_description=current.short_description;
2.3 grinc.caller_id=current.caller_id;
2.4 grinc.parent_incident = current.sys_id;
2.5 grinc.work_notes = 'this incident is copied from'+ current.number;

2.6 grinc.initialize();
2.7 grinc.insert();


action.setRedirectURL(grinc);
```

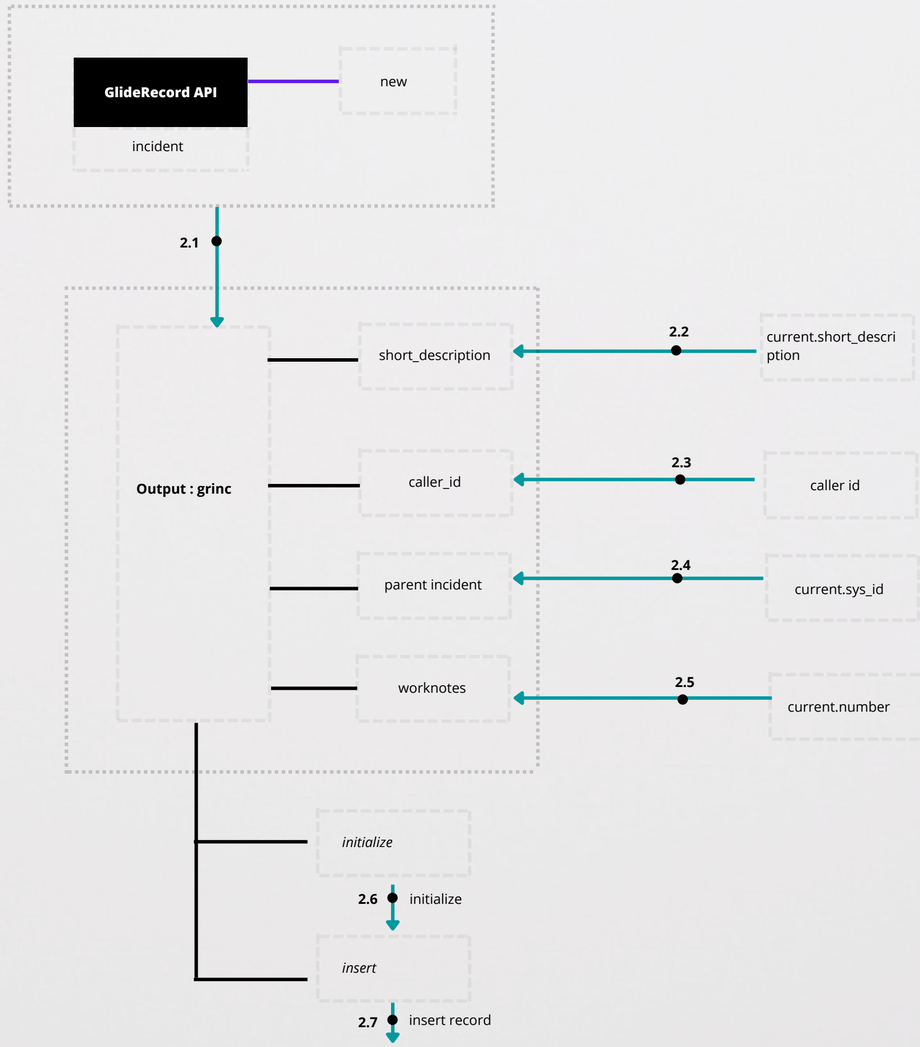
3 Parent Incident

Child Incident

2.1 creating **grInc** a **GlideRecord Object** of the incident table

2.2 current represent the incident table and current.short description, is the current incident record short description, store this value in the grInc object more spefically in the the short_description variable.
use the same analogie to store caller id, sys_id and also work notes, respectively in step **2.3, 2.4, 2.5**

2.6 initialize the grInc object and **2.7** insert a new record



14

Update the incident priority to critical with an UI Action

Create an ui action in the incident table, give it a name, choose type form button and select show insert and update also client, create function setPriority() ;add this function to the Onclick field.

Update Incident Priority

1 User clicks on the ui action

2 calling the onClick function **setPriority function**

The ui action script is written on the incident table

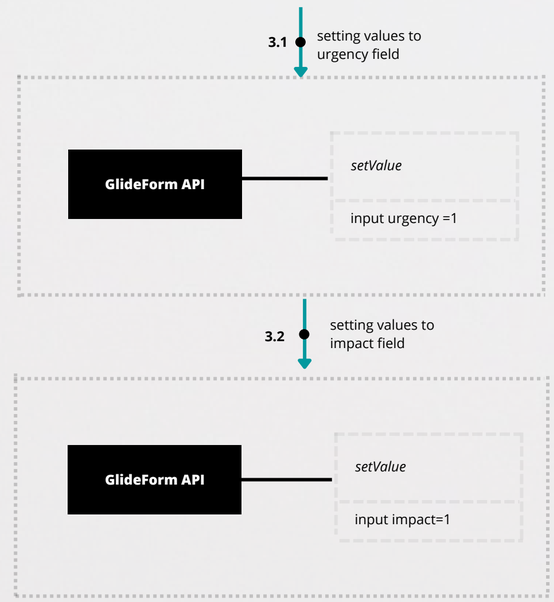
3
```
function setPriority(){

g_form.setValue('urgency',1);
g_form.setValue('impact',1);

}
```

Urgency: High
Impact : High
Priority ; High

4

3.1. Use GlideForm API and setValue method to set urgency field value to 1

3.2. Use GlideForm API and setValue method to set impact field value to 1



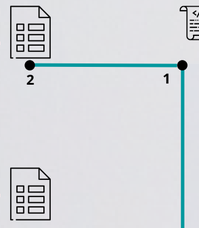
15

Create a Knowledge Base article using an UI macro

UI macros are scripted components in servicenow that admins can add to the user interface.

The ui macro name is create_kb, which is a database name, mention it in the attribute field, attribute is a dictionary field of caller reference field.

Caller Reference Field dictionary fields form
Attributes : **ref_contributions=create_kb**



Incident form
Caller Reference Field create kb **ui macro button available next to the field**

UI Macro script :
kb_create

```
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core"
xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">


<button id="button" onclick="kbCreate();">Post
Knowledge</button>


<script>

 function kbCreate(){
 var grkb= new GlideRecord('kb_knowledge');
 grkb.initialize();
 grkb.short_description =
g_form.getValue('incident.short_description');
 grkb.text=g_form.getValue('incident.description');
 grkb.insert();
 }

 </script>


</j:jelly>
```

The ui macro here is button visible to the caller reference field, to make it visible, all you need to do is to create a ui macro, note down the ui macro database name and navigate to incident caller reference dictionary field and the following line : **Attributes : ref_contributions=create_kb**

This will automatically create a ui macro button, now add more functionality to it, here we would like to create a knowledge base article by retrieving some info form the incident form.

1.1 Begin by creating xml and jelly tag.

1.2 Create html button and create kbCreate function which will be called on a click.

1.3 inside the script tag, create the kb create function

1.4 Create a GlideRecord Object of kb_knowledge table,

1.5 initialize the object

1.6 Assign object a value for the object short description, it will be the incident short description,

1.7 for text field, assign incident description

1.8 Insert a record

16

Update the incident priority to critical with an UI Action and save the record (client and server side script)

Create an ui action in the incident table, give it a name, choose type form button and select show insert and update also client, create function updateToPriority() add this function to the Onclick field.

Update Incident Priority

1 User clicks on the ui action

2 calling the onClick function **updateToPriority**

UI Action client / server Script

The ui action script is written on the **incident table**

3
```
function updateToPriority()
{
alert('you are going to change the priority');
```
3.1 `gsftSubmit(null,g_form.getFormElement(),'update_to_priority');`
```
}


```
3.2 `if ( typeof window=="undefined")`
3.3 `setPriority();`
```
function setPriority()
{
```
3.4 `current.urgency='1';`
3.5 `current.impact ='1';`
3.6 `current.update();`
```



action.setRedirectURL(current);
}
```

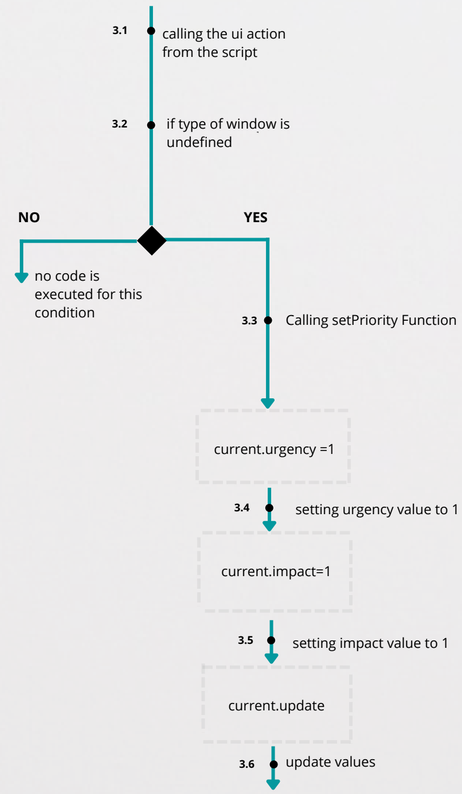
4 Urgency: High
Impact : High
Priority ; High

In this script the ui action is called twice, first when the user clicks the ui action and for the second time, the code itself call the ui action

3.1 Calling the ui action from the script, with the method gsftSubmit which gets as input the update_to-priority, the database name for the ui action

3.2 if the type of window is undefined, we can call a server side code otherwise not

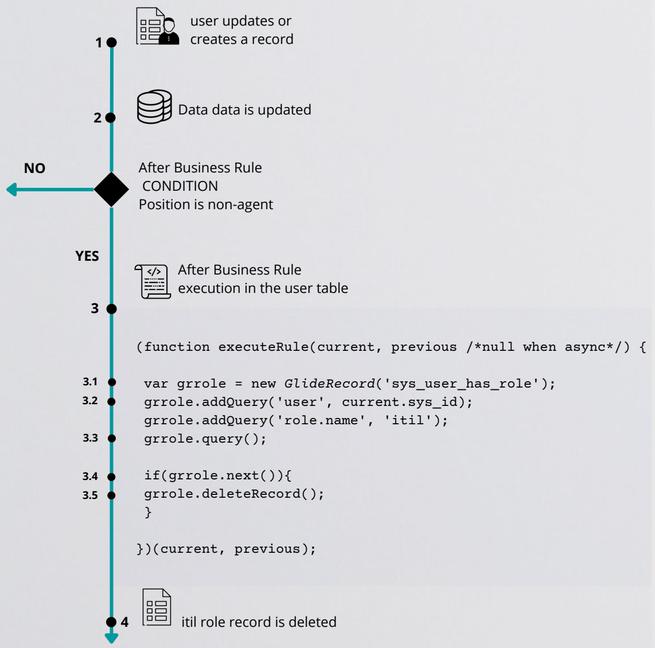
3.3 In the function set priority, **3.4** setting the impact and **3.5** urgency value to 1. **3.6** in the end update the current record



17

If a user in his profile has position equal to non-agent, the user has ITIL role, remove ITIL role for the user.

This requirement is a verification of field values and action taken based on the value, therefore, write an after business rule, for updates and insert. This br will have in condition : postion is equal to non-agent, run a script which will look for the current user and verify if the user has ITIL role, if yes it will remove the role by deleting the record.



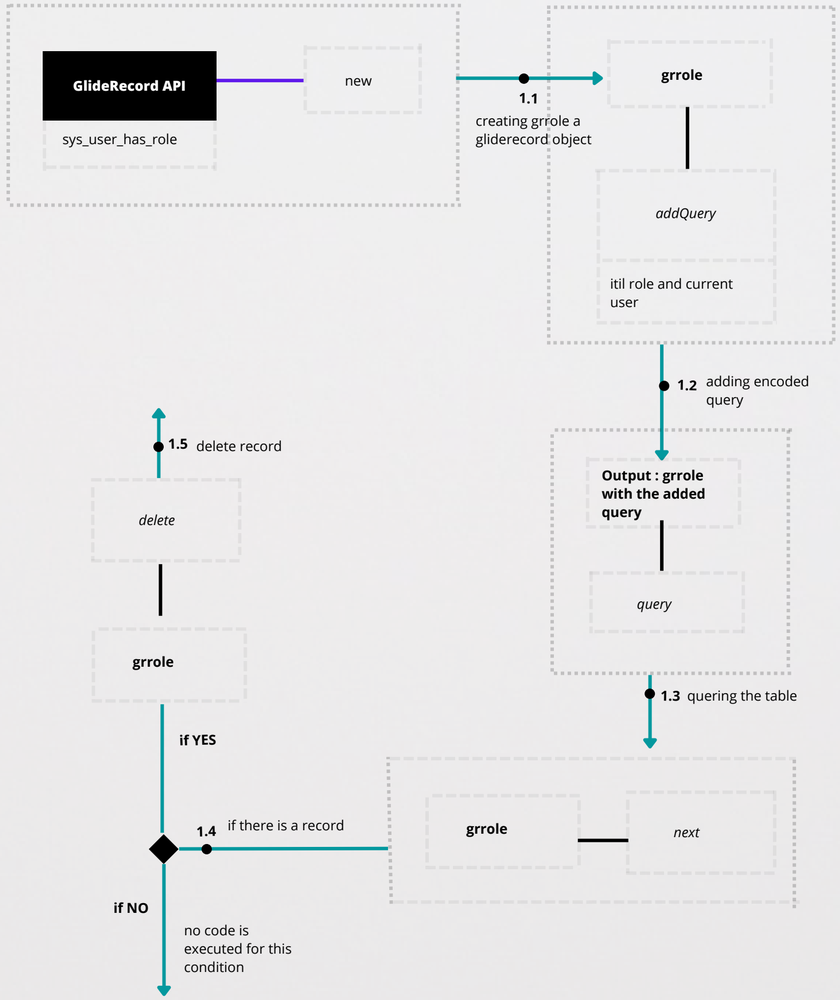
1.1 Create a GlideRecord Object of sys_user_has_role table, the objet is grrole.

1.2 Add query, the query is the user is the current user sys id an d role is itil.

1.3 Query with grrole object

1.4 if there is record with the condition mentioned previoulsy,

1.5 delete the record



18

Create 2 problem task linked to one problem

Create an ui action in the problem table, give it a name, choose type form button and select show insert and update.

create 2 problem task

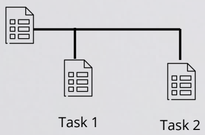
1 User clicks on the ui action

UI Action Script, it is written on the problem table

2

```
2.1 var des =[“Test 1”, “Test 2”];

2.2 for(var i=0;i<des.length;i++){
2.3 var grpb = new GlideRecord(’problem_task’);
    grpb.initialize();


2.3 grpb.short_description=des[i];
2.4 grpb.description=current.short_description;
2.5 grpb.problem = current.sys_id;
2.6 grpb.problem_type=’General’;
2.7 grpb.insert();

    }


    action.setRedirectURL(current);
```

3 Problem record



2.1 Begin by creating an **Array** which contains two values test 1 and test 2. the name of the array is des.

2.2 Run a for loop for the array's length, for each iteration exercute the upcoming steps.

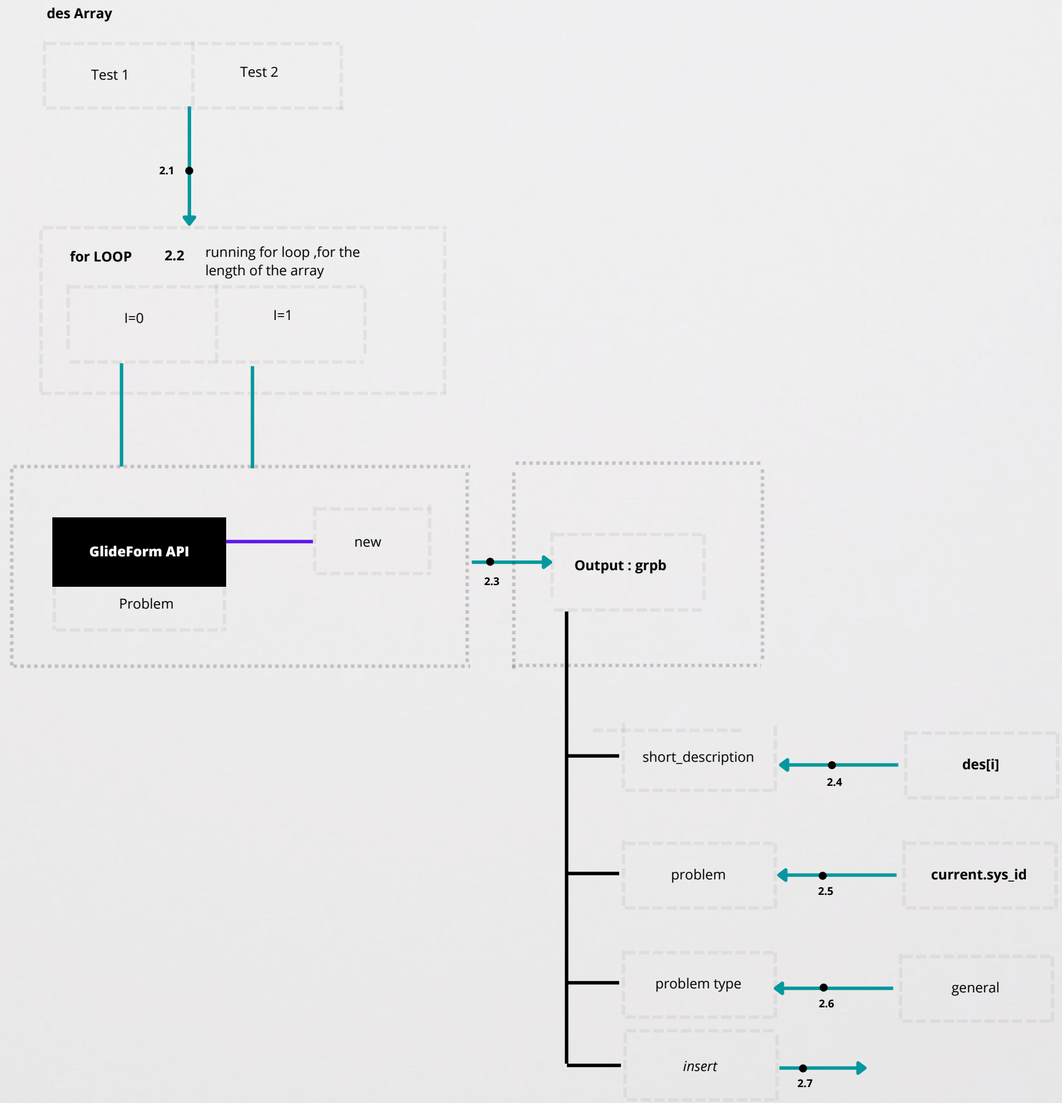
2.3 Create a **grPb** a GlideRecord object of problem_task table

2.4 current represents the problem table, current.short-description is the current problem table record short description , store that value in the grPb object, exactly in grPb.description.

2.5 Do the same thing for problem sys id and **2.6** assign the value general to problem type.

2.7.Insert the record.

As the loop will run twice, it will insert two records.



19

Group incidents by assigned_to and list only grouped records where the assigned_to have more than two records

Test the script in background script to see the result

Background script

```
1.1  var agg = new GlideAggregate('incident');
1.2  agg.addAggregate('COUNT', 'assigned_to');
1.3  agg.query();
1.4  var arr = [];


1.5  while (agg.next()) {
1.6  var incidentCount = agg.getAggregate('COUNT', 'assigned_to');


1.7  if (incidentCount > 2) {

        var inc = new GlideRecord('incident');
        inc.addEncodedQuery('assigned_to=' + agg.assigned_to);
        inc.query();

    while (inc.next())
     arr.push(inc.sys_id.toString());
    }}

    return arr;
```

1.1 Use GlideAggregate API to create **agg object** of incident table.

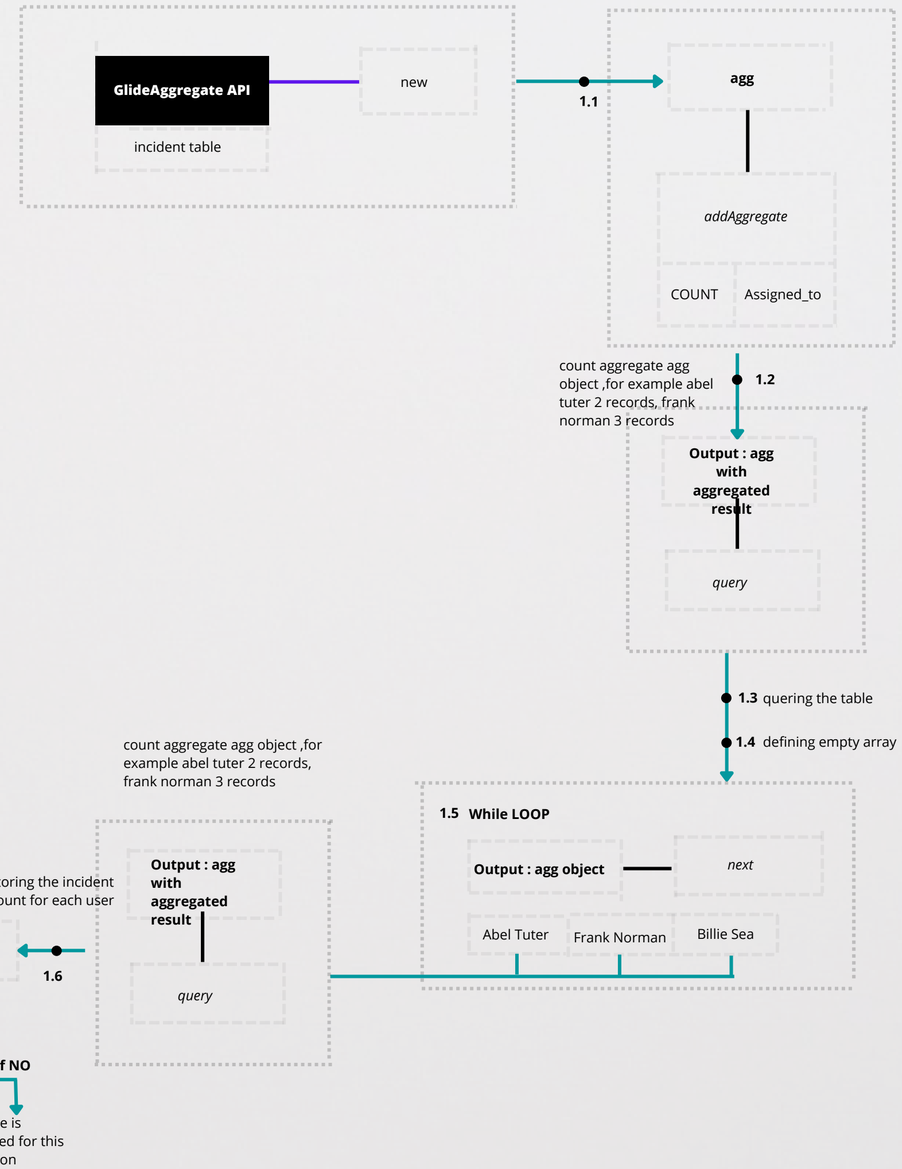
1.2 Use **addAggregate** method to **aggregate** the result by assigned it will **return the number of records** assigned for each user.

1.3 query with agg object to have those results

1.4 create an empty array.

1.5 create a **while loop**, in the loop as **a first step**, get the **agrregated count of incidents** and **1.6** store in the **incidentCount** variable.

1.7 now compare the **incidentCount value to 2**, if it is more than two, than means, that particular use have more than two records.



20

Group incidents by assigned_to and list only grouped records where the assigned_to have more than two records

Test the script in background script to see the result

Background script

```
var agg = new GlideAggregate('incident');
agg.addAggregate('COUNT', 'assigned_to');
agg.query();
var arr = [];



while (agg.next()) {
var incidentCount = agg.getAggregate('COUNT', 'assigned_to');



if (incidentCount > 2) {

    var grinc = new GlideRecord('incident');
    grinc.addEncodedQuery('assigned_to=' + agg.assigned_to);
    grinc.query();

while (grinc.next())
 arr.push(grinc.sys_id.toString());
}}

return arr;
```

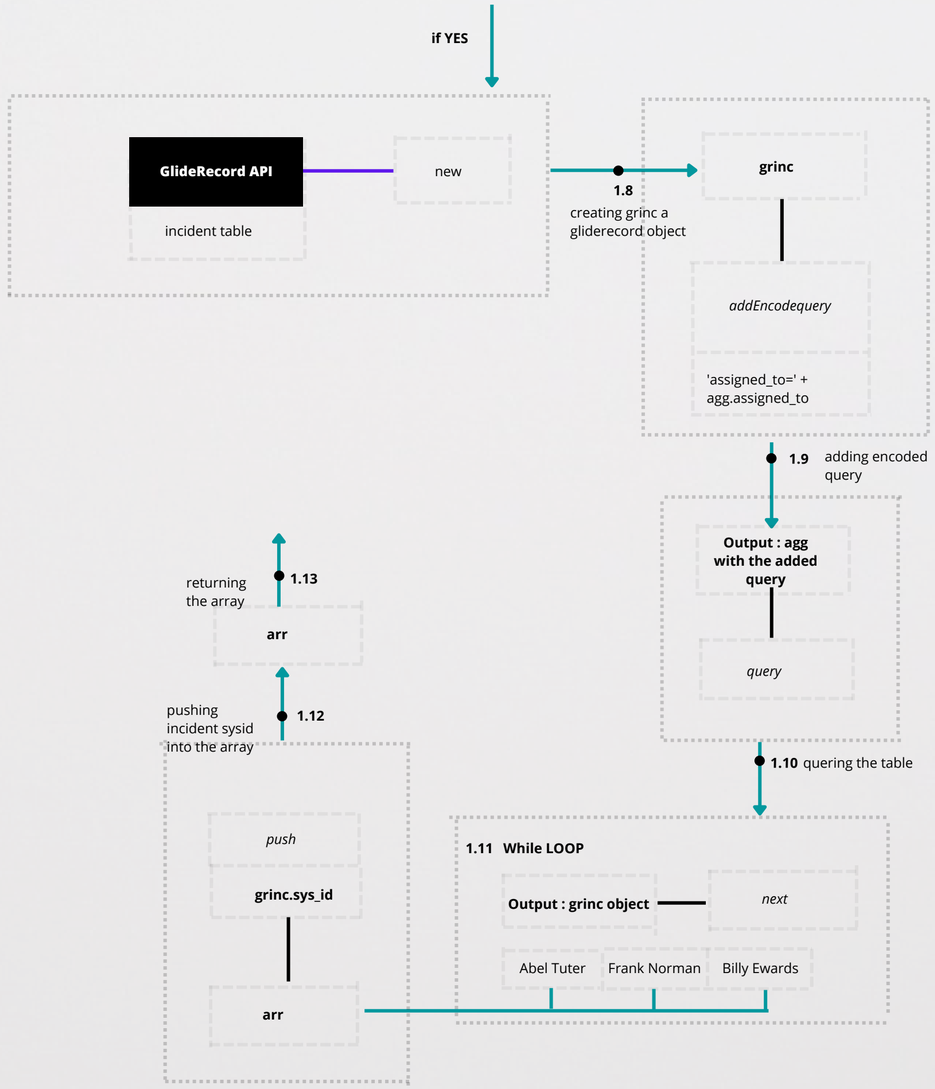
At this point, we have a user who have more than two records, and now we write the code to group all incidents belongs to the user and put it an array, we have already created an empty array, we will use that.

1.8. we remain in the if condition, create a grInc a GlideReocrd object of the incident table, **1.9** add a encoded query , filter by assigned to , use agg object the get the assigned to value.

1.10 query.

1.11. create a while loop, for each incdent found in the loop, push the sys id to the arr empty array. the while loop will end once we are with alll filter incidents.

it seems like we have many while loops and a condition. we can group all incidents which belongs to an user and push the result into an empty array without writing the first loop, but the first loop is necessary to make sure that the user has more two records, therefore we need it, check that info in a condition and trigger the second while loop for grouping records.



21

If a user in his profile has position equal to agent, add ITIL role to the user.

This requirement is a verification of field values and action taken based on the value, therefore, write an after business rule, for updates and insert. This br will have in condition : postion is equal to agent, run a script which will get the current user and add itil role.

1 user updates or creates a record

2 Data data is updated

After Business Rule Condition : Position is agent

NO

YES

After Business Rule execution in the user table

3

```
(function executeRule(current, previous /*null
when async*/) {

var role = new GlideRecord('sys_user_has_role');
role.initialize();
role.user = current.sys_id;
role.role= "282bf1fac6112285017366cb5f867469";
role.insert();
})(current, previous);
```

4 itil role record is created

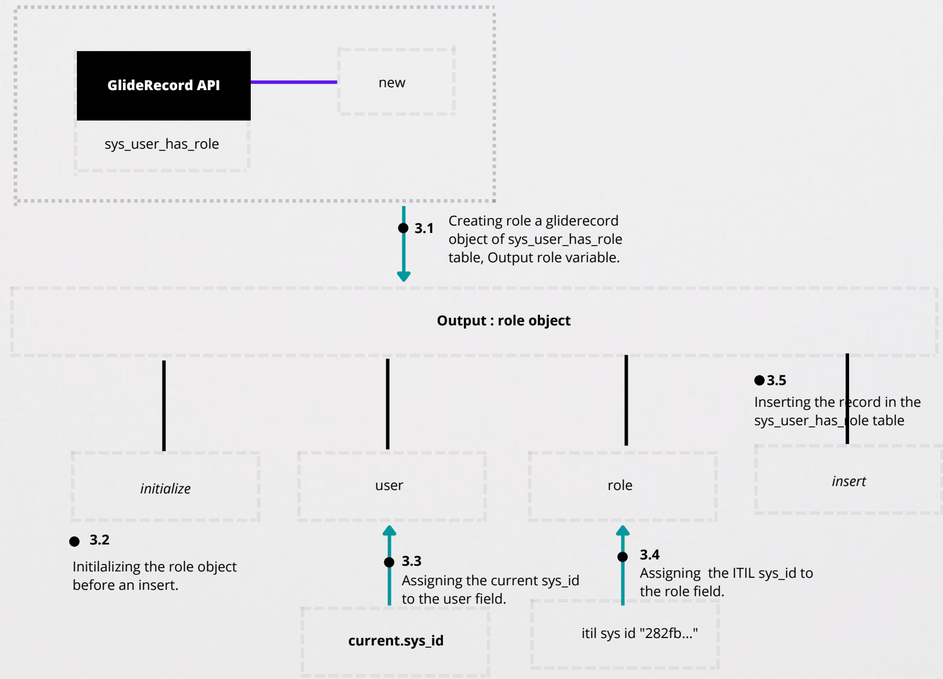
1.1 Create a GlideRecord Object of sys_user_has_role table, the objet is role.

1.2 initialize the role object

1.3 use role object to assign a value for the user field, assign the current user sys id

1.4 use rold object to assign itil sys id to role field

1.5 insert a new record.



22

Send a notification to the assigned to of an incident , whenever a custom date field is superieur than the current date time.

Register an trigger.notification event for the incident table, create as well a beta notification triggerd when the event is fired, then write a scheduled script to query the incident table for meeting the condition above, if there are records overdue 30, 60, 90 days , generate overdue reminder from the script, which will automaticatlly send the notifications.

Scheduled script
for every hour

1

```
var grInc = new GlideRecord('incident');
grInc.addActiveQuery();
grInc.query();
while(grInc.next()){
if(new GlideDateTime(gr.reached_date).getNumericValue()
>= new GlideDateTime().getNumericValue()){


 gs.eventQueue('trigger.sendoverdueremindernotification'
,grInc,grInc.assigned_to.toString());
}}
```

3 notification
send overdue reminder
is executed

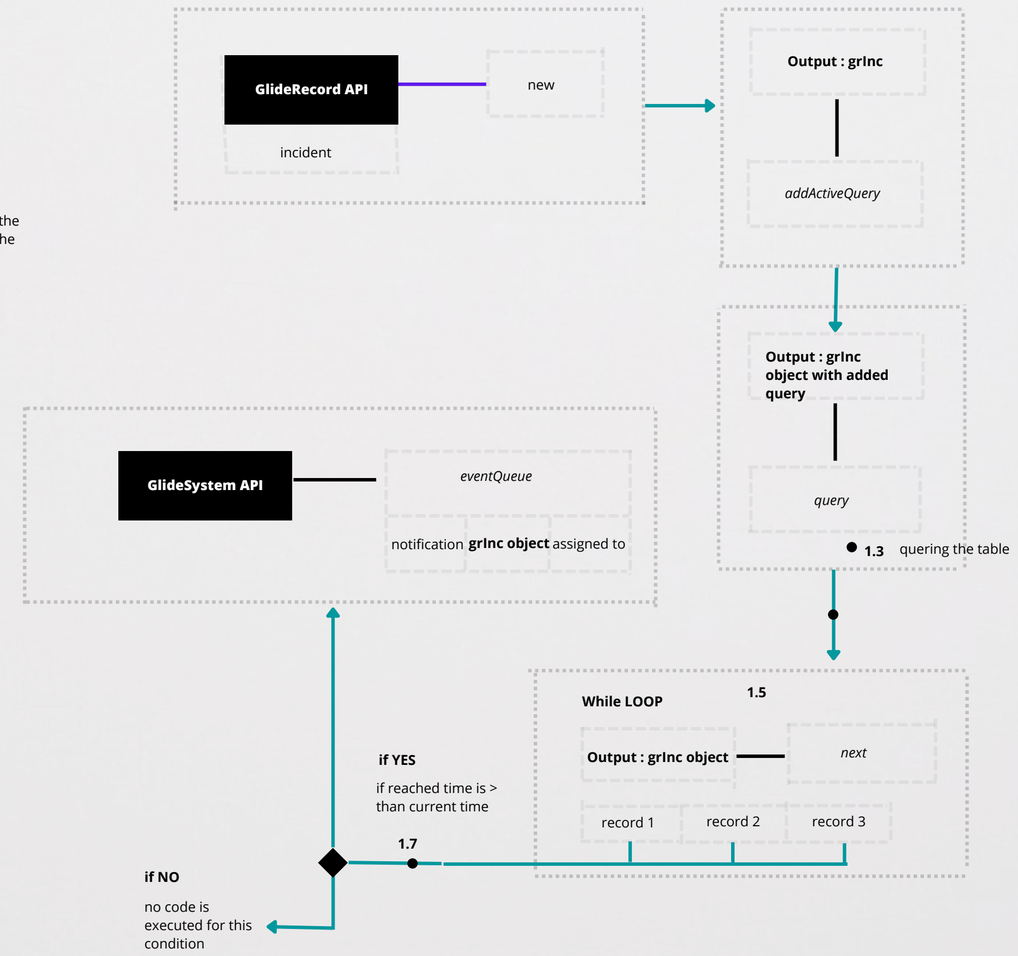
1.1 create grInc a GlideRecord object of incident table,

1.2 add query to retrieve only active records

1.3 query the table

1.4 for each record found , get the reached date time numerical value and compare ti to current date time numerical value, if it it is superior , execute the code further.

1.5 queue an event which is already created, mention the assigned to with the grInc object in the paremeter, so the notification will be sent to that person.



23

In the Incident form make the contact type field editable only for Assignment group members.

For this requirement use a scratchpad object which will have a boolean value either true or false, if the current user is not member of that particular assignment group it will be false, based on the scratchpad value, modify field attributes. write a display Business Rule on Insert and update, which will have a scratchpad object, the value it will be true if the logged in user is member of the assignement group, use the scratchpad on the client side, if the value is false, meaning the user is not member then you set the field read only, the opposite condition will let the user access the field.

1 User opens a incident record

2 Data loads from the database

A Display Business Rule can instantiate **g_scratchpad** object which can be passed to the client side and it can be used in client script.

3

```
(function executeRule(current, previous /*null when
async*/) {

g_scratchpad.isMember =
gs.getUser().isMemberOf(current.assignment_group.toString()
);

})(current, previous);
```

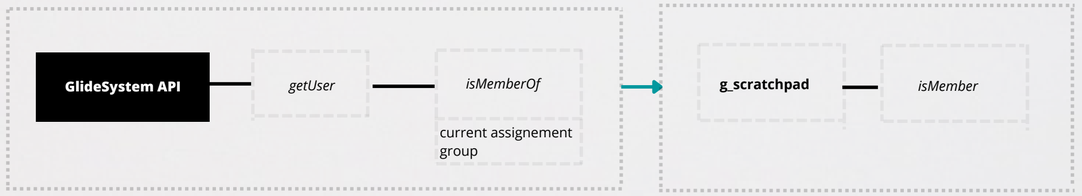
Client script execution
The client script is written for the **incident** table.

1

```
function onLoad(){

 if(g_scratchpad.isMember.toString() == 'false'){
  g_form.setReadOnly('contact_type', true);
  // add few more fields as per your requirement
 }

}
```

2 contact type is editable for group members

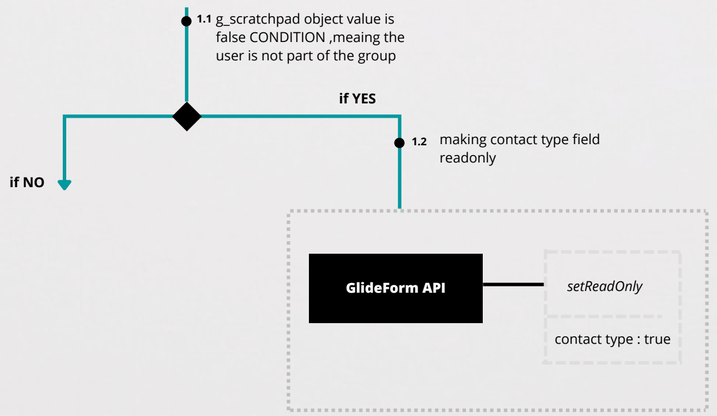
3.1 Create a Display Business Rule for the incident table, use the script which checks if the current user is part the assignment group and stores the value in a scratchpad object g_scratchpad.isMember



Server Code

1.1 Create a onLoad Client script on the incident table use the scratchpad object in a onload client script if the value is false, meaning the user is not part of the group, make the contact type read only with setReadOnly method

Client Code



24

Add 30 seconds to the current date/time and set this value to a custom date time field from an UI Action.

create an ui action, it will have a script which will get the current time, add 30 seconds and then the new value to the custom date field, the script is called whenever you click on the ui action.

Add 30 seconds

1 User clicks on the ui action

2 UI Action Script, it is written on the incident table

```
2.1 var gdt = new GlideDateTime();
2.2 gdt.addSecondsUTC(30);
2.3 current.published_date= gdt;
2.4 current.update();
    action.setRedirectURL(current);
```

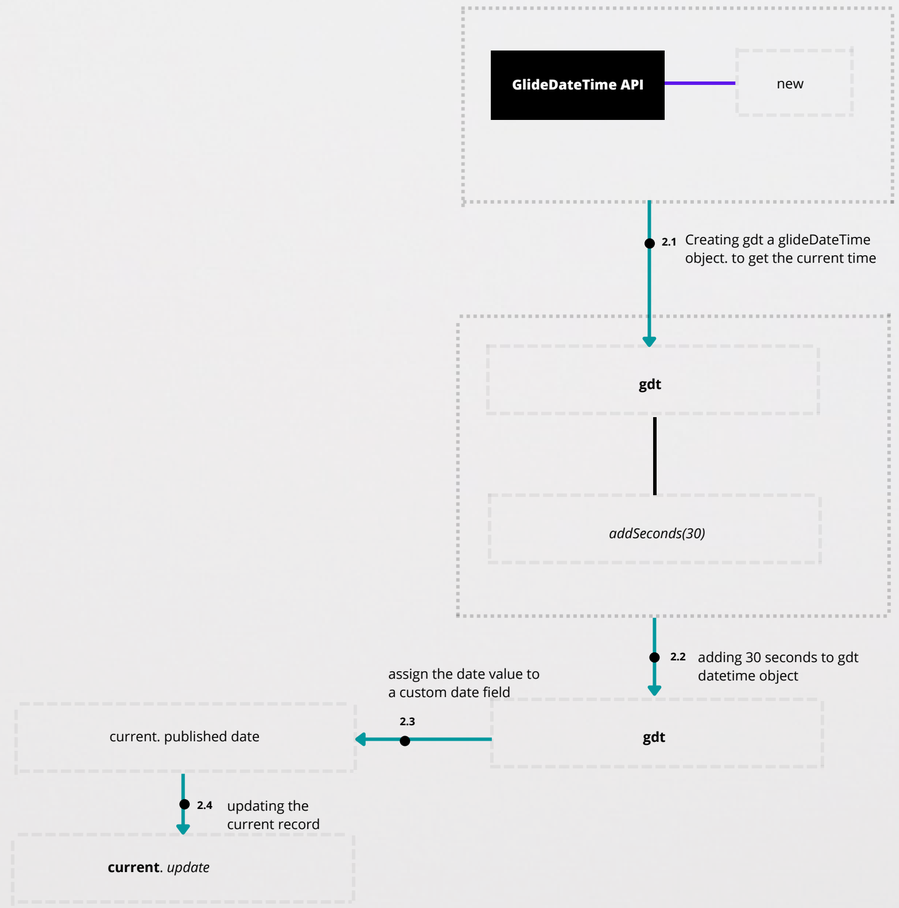
3 Incident record
published date : 2022-12-05 : 15:25:15

2.1 create a glidedatetime object, the object is gdt

2.1 user addSeconds method, add 30 second the current date time object

2.3 assign this value to the current purblished date filed

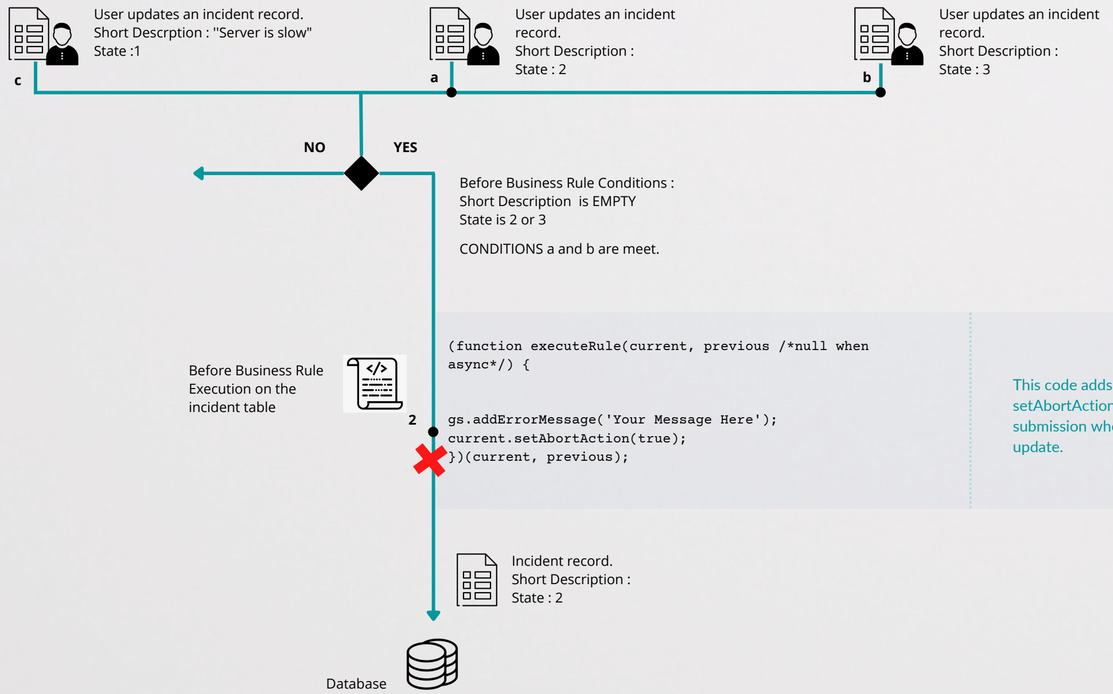
2.4 update the record and redirect the user to the current record



25

Restrict Submission of an incident form when the description field is empty and the state field value is 2 or 3.

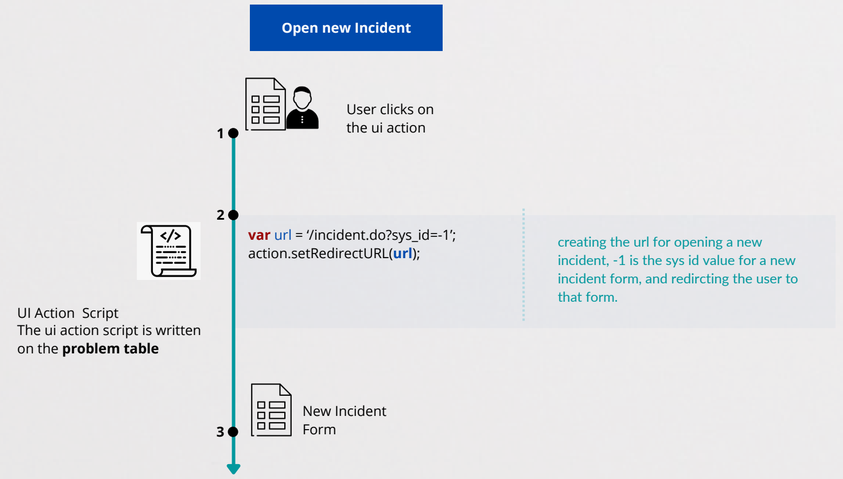
Write a before business rule and the script to abort the action, this br is written on insert and update.



26

Open a empty incident form from an problem record

Create an ui action in problem table, give it a name, choose type form button, and select show insert and update.



27

Hide an icon of the incident form.

There is an icon of the form for more options, hide that icon when the form loads.

1 User opens an incident record

2 Data loads from the database

3 Client script execution
The client script is written for the **incident** table.

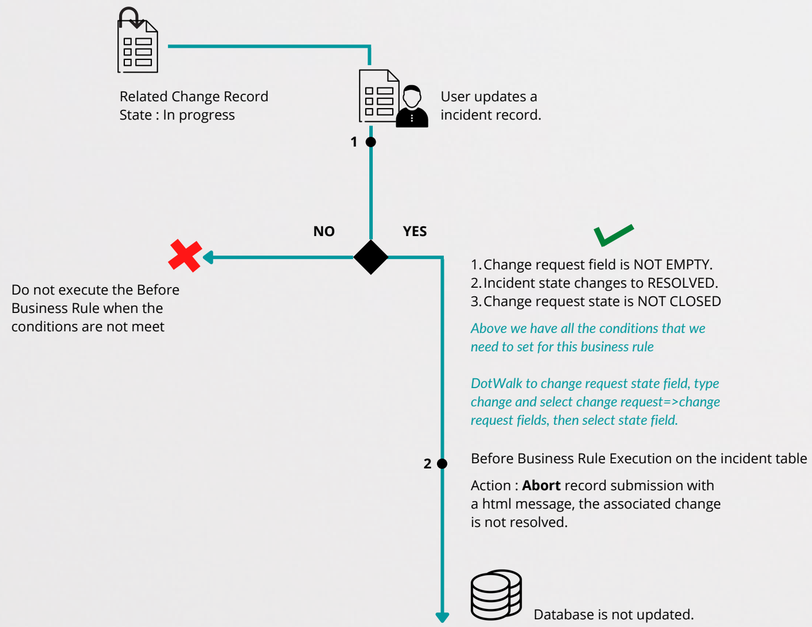
```
function onLoad() {

var icon=document.getElementById("toggleMoreOptions");
icon.style.display='none';
}
```

Get the HTML element of the icon and store it in a variable. Change css display style to none.

4 3 dotted icon is **not visible**

28

Abort the submission of an incident in resolved state if the related change record is not complete.



29

When the CI changes in the incident, update subquently the description with asset tag, and company name

Configuration Item : *BUD-IBM

1 User or system changes the CI field value, and the CI is now *BOW-IBM

Client script is written on the **Configuration Item** field

2

```
function onChange(control, oldValue, newValue,
isLoading, isTemplate)
{

if (isLoading || newValue === '') { return; }


var ga = new GlideAjax('CIUpdates');
ga.addParam('sysparm_name', 'getCIDetails');
ga.addParam('sysparm_ci', g_form.getValue('cmdb_ci'));
ga.getXML(showDetails);
```

/ Instantiate **CIUpdates** Script Include with GlideAjax, add **getCIdetails** method and **cmdb ci** field value as parameters.*
*Use getXML method to make a **server side call** **CIUpdates().getCIdetails()** along with cmdb ci value as an input parameter.*
*The result will be returned to **showDetails** callback funcion. */*

```
var CIUpdates = Class.create();
CIUpdates.prototype = Object.extendsObject(AbstractAjaxProcessor, {


getCIDetails:function(){
var grci = new GlideRecord('cmdb_ci_computer');
grci.addQuery('sys_id',this.getParameter('sysparm_ci'));
grci.query();
```

/ Create grci a **GlideRecord object** of cmbd_ci_computer and **query** cmbd_ci_computer table with the cmbd_ci value, */*

with sysparm_ci we get the cmbd ci value.

Asynchronous Call
Client / Server script execution

```
if (grci.next()){
var details = "asset tag:" + grci.asset_tag + "\n" + "company:" +
grci.company.name;
return details;
        }
            },

type: 'CIUpdates'

});
```

/ If there is a cmdb record, assign tag field value and company name field value to the details variable, **return** details values. */*

```
function showDetails(response) {

var answer =
response.responseXML.documentElement.getAttribute("ans
wer");
g_form.setValue('description', answer);

}

}
```

/ response variable holds the result in XML format, results are available in the answer Attribute, which will retrieve and assign those values to the description field. */*

3 Configuration Item : *BOW-IBM
Description : 'Asset tag : P1000230 company : ACME North America

30

In the story form, if the state is in progress and acceptance criteria is not empty, make accetpance read only

UI policy :,Make Acceptance Crtiteria read only.

When to Apply : When STATE is in Work In Progress AND Acceptance Criteria IS NOT EMPTY

- onload
- global
- reverse

onload, global and reverse is false are checked

UI policy Action : On the acceptance criteria field.Make Read only is True, visible and Mandatory are leave only.

If there are mulitple ui policies for the same field, change the order to solve this issue.