**Business Rule**

**Scenario 1 : If the Problem Priority = ‘Critical’ then create “General” and “Root Cause analysis” Problem task.**

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

    // Add your code here

    var type = ['general','rca'];

    for(var i=0;i<2;i++){

        var gr = new GlideRecord('problem\_task');

        gr.initialize();

        gr.short\_description = 'Task type is '+type[i];

        gr.problem\_task\_type = type[i];

        gr.problem = current.sys\_id;

        gr.insert();

    }

})(current, previous);

**Scenario 2 : Write a functionality to transfer the work notes from Incident to associated Problem.**

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

    // Add your code here

    var gr = new GlideRecord('problem');

    gr.addQuery('sys\_id',current.problem\_id);

    gr.query();

    if(gr.next()){

gr.work\_notes = current.work\_notes.getJournalEntry(1); // it will return recent one

//gr.work\_notes = current.work\_notes.getJournalEntry(-1); // it will return all work notes

    }

})(current, previous);

**Scenario 3 : Write a functionality to count the Associate Incident.**

**(To do: adjust the count if the Associate Incident gets deleted)**

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

    // Add your code here

    var gr = new GlideRecord('problem');

    gr.addQuery('sys\_id', current.problem\_id);

    gr.query();

    if(gr.next()) {

        gr.u\_associate\_integer\_count = gr.u\_associate\_integer\_count + 1;

    }

    gr.update();

})(current, previous);

**Scenario 4:**

**Whenever an Incident is created with Priority = 1 (Critical), it must:**

**Automatically assign to the On-Call Support group.**

**Send an email notification to the Incident Manager.**

BR - before insert

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

if (current.priority == 1) {

current.assignment\_group = 'PUT-YOUR-GROUP-SYSID-HERE';

current.assigned\_to = 'PUT-YOUR-USER-SYSID-HERE';

current.work\_notes = "Auto-assigned due to Critical priority";

}

})(current, previous);

Steps to Configure Your Event

✅ 1. Create Event in Event Registry

Navigate: System Policy → Events → Event Registry

Click New

Fill in:

Name: incident.critical.notify (exactly what you used in script)

Table: Incident [incident]

Application: Global (or your scoped app)

Description: Event to notify Incident Manager for Critical incidents

Save

✅ 2. Create a Notification

Navigate: System Notification → Email → Notifications

Click New

Fill in:

Name: Critical Incident Notification

Table: Incident [incident]

When to Send: check Event is fired

In the Event name field → select your event: incident.critical.notify

Who will receive:

Add recipient (e.g., Assigned to, Incident Manager, or specific group).

What it will contain:

Add subject/body for the email.

Example subject: Critical Incident: ${number}

Example body:

A critical incident has been created.

Number: ${number}

Short description: ${short\_description}

Assigned to: ${assigned\_to}

**Scenario 5: Set SLA Automatically**

* **Business Requirement**:  
  When a **Change Request** is created with **Type = Emergency**, automatically attach the *Emergency SLA*.
* **Why?**  
  Emergency changes are high-risk and need faster handling. This ensures SLA tracking starts immediately without manual action.

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

// Check if Type is Emergency

if (current.type == 'emergency') {

// Attach Emergency SLA

var sla = new GlideRecord('contract\_sla');

if (sla.get('name', 'Emergency SLA')) {

var slaUtil = new SLAUtil();

slaUtil.startSLA(current, sla.sys\_id);

}

}

})(current, previous);

***Note*** – Go to SLA table and crate **Emergency SLA** and setup it

**🔹 Scenario 6: Prevent Deletion of Active Change Request**

* **Business Need**:  
  You want to **stop users** from deleting a Change Request if it’s in **Implement state** (or any important state).
* **Why?**
  + Avoids accidental or unauthorized deletion of important records.
  + Keeps the audit trail intact.

(function executeRule(current, previous) {

// Check if Change Request is in Implement state

if (current.state == 'implement') {

// Stop the deletion

gs.addErrorMessage("You cannot delete a Change Request in Implement state.");

current.setAbortAction(true);

}

})(current, previous);

**Scenario 7: Send Notification on Assignment Change**

* **Business Need**:  
  When an **Incident’s Assigned To field** changes, the new assignee should receive an **email notification**.
* **Why?**  
  Keeps assignees updated instantly when work is handed over.

(function executeRule(current, previous) {

// Check if Assigned To changed

if (current.assigned\_to.changes()) {

// Trigger event with incident info

gs.eventQueue(

"incident.assignee.changed", // custom event name

current, // GlideRecord (incident)

current.assigned\_to.toString(), // new assignee sys\_id

current.number.toString() // incident number

);

}

})(current, previous);

**Scenario 8:** You have a **Business Rule** that needs to run **after an Incident is updated**. The requirement is: *if the Incident’s priority is changed to "1 - Critical", automatically add a work note saying "High priority! Notify support team."*

** ***When:*** *after (because we want it to run after the record is updated)*

** ***Insert:*** *false*

** ***Update:*** *true*

** ***Filter conditions:*** *Priority changes to 1 - Critical*

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

    // Add your code here

    if (current.priority == '1' && previous.priority != 1) {

      var gr = new GlideRecord('incident');

        if (gr.get(current.sys\_id)) {

            gr.work\_notes = "High priority! Notify support team.";

            gr.update();

        }

    }

})(current, previous);

 We create a new GlideRecord object to update the same record.

 **Why?** In an **after BR**, using current.update() can cause recursion. GlideRecord avoids this problem.

 gr.get(current.sys\_id) → fetches the same Incident record.

 gr.work\_notes = ... → adds the work note.

 gr.update() → saves the change.