

servicenow

DEVELOPment

{BY}

SRINIVASA SUNKARA

MYTHRI XEROX

CELL NO.7995810201

About Author

Srinivas Sunkara is **ServiceNow Architect & Solution Provider** and founder of **Dettifoss IT and Gautham IT Solutions Pvt.Ltd.** The education background includes in Master of Computer Application (**MCA**) from Acharya Nagarjuna University, Guntur, India. Total **11+ years** of IT expertise and career was begin as Lotus Notes Developer. Previously worked on various technical platforms like... Share Point, Oracle Business Intelligence, Robotic Process Automation and past four years into Service now in software development. Training has been my passion and almost given training for **65,000+ Professionals**.

I would like to thanks my spouse **Satyavani** and son **Gautham** given great support to end this first edition of **Service Now LIVE book** and allowing to work at late night and I really appreciate their patience, way of understand, love, support, helping

I enjoyed meeting new people and hearing new perspectives. Reach out if want to talk to me about emerging technology called service now. Creating magic's with our platform. I focus on making new high-quality professional to serve our productivity

I have spent last several years utilizing my professional background as training experts and my service deliver to entire world. Currently lives in Hyderabad, India, working with **Dettifoss IT & Gautham IT**

Content Editors & Digital marketing

Special Thanks to

Mr. Sai Kumar Pothuguta Mr. Raghu Ram Allamsetty

Srinivas Sunkara

Why I am writing this book

I have been working since **5+Years** on service now platform into various applications and modules and deep look into the platform. These all days I faced lots of struggles but did not find any solutions. There is no step by step guide available in market. Finally, I decided to write step by step instruction servicenow Development guide that will help to people who already working on platform also for new learners

My Service Now LIVE book Development will provide step by step instructions in topic wise and very crystal clear, New learners and student are felling happy with my book We have worked on this practice **Service Now LIVE book**, each topic is categorized into simplified very practical steps that we use with many customers and students. Our approach has improved practice make it easy in this book we will learn how to get hands on experience on each applications and modules. It was mainly designed for everyone who wants to become a master in service now with practical exercises

Srinivas Sunna

This book for Whom

Who wants to become an expert in **servicenow** platform like Admins, Developers, implementors and architect's ant they will expect each topic is step by step guide lines and very clear with screen shots Who begin recently to learn service now application and modules and who already working in organization, still anybody looking for step by step hand guide. **This book is help them 100%**

Srinivas Sunkara

ServiceNow Live Book

Second Edition (Development)

Managing Editor: **Srinivas Sunkara**

Editorial Support:

Mrs. Satyavani Bandarupalli

Mr. Syed

Mr. Raghavendra

MR. Sai Babu Pothugunta

MR. Raguram Allamsetty

Cover Page Design & Illustrations: **Mr. Sai Babu Pothugunta, Mr. Raghu Ram Allamsetty @ 2021 C.P. Dettifoss IT (ServiceNow academy)**. All Rights Reserved. Although every care has been taken to avoid errors and omissions, this publication is being sold on the condition and understanding that the information given in this book is merely for reference and must not be taken as having authority of or binding in any way on the authors, editor, publisher or sellers.

Neither this book nor any part of it may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, microfilming and recording or by any information storage or retrieval system, without prior permission in writing from the copyright holder.

Trademark notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe. Only the publishers can export this book from India. Infringement of this condition of sale will lead to civil and criminal prosecution.

First Edition: 2019 Aug, printed in India

Published by

Dettifoss IT & Service Now Acaademy

Office Address

Flat no: 302, 3rd Floor, Namdev Block-A, Prime Hospital Lane, Ameerpet,
Hyderabad, Telangana.

Website: www.dettifosst.com

India contact: +91-9949429009

Lesson-01

Glode API & Glide Record

Agenda

- Glide API Overview
- Types of Glide API's
- What is Glide Record and Usage
- Glide Record Architecture
- API Mapping
- Glide Record Methods
- Glide Record Exercises

Srinivas

Glide API Overview?

Service now Developer often using **Glide API** in now platform to change default behavior of the application and customize existing functionality.

Glide Classes are providing more flexibility to interacting with snow application through scripting. Using by Glide API's we can perform some database operation instead of writing any **SQL Queries**

Each API contain lot of methods and each method perform different operations in Service Now Applications

Types of Glide API's

Glide API's

Client Side	Server Side
Glide Form	Glide Record
Glide User	Glide System
Glide Ajax	Glide Date
Glide Dialog Window	Glide Date and Time
Glide List	Glide Aggregation
Glide Menu	Glide Element

What is Glide Record and usage?

This is most common and important API and frequently using in now platform. Glide Record is special Java class its native java script class and mainly running from **Server Side**. Glide Record is used to perform CRUD operation instead of writing **SQL Queries**. This API handle both **rows** and **columns** in underlining database.

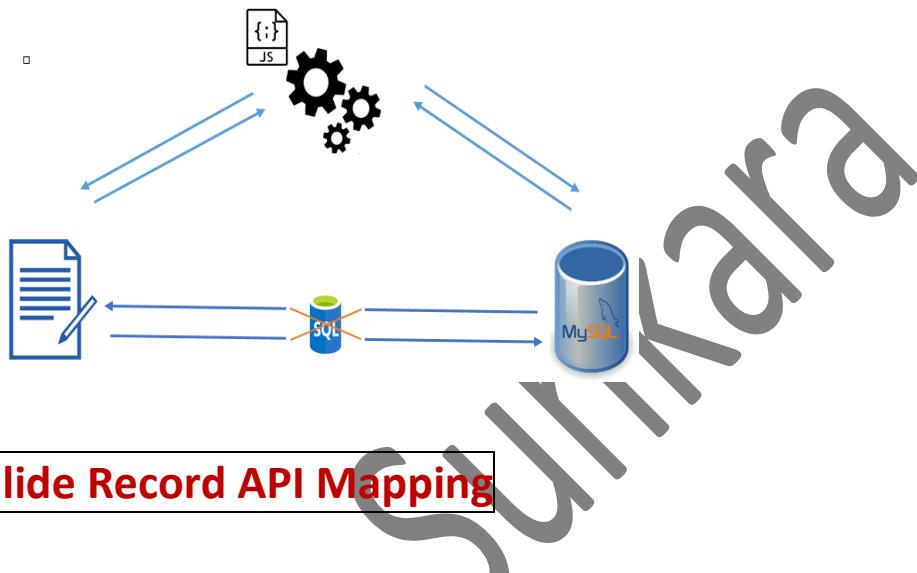
In service now platform directly we can't interact with database to perform any CRUD with **SQL Queries** operations if we want to do this activity then we can go for **Glide Record**.

Note: Always we need to test queries on a non-production instance before deploying them into production environment. An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query. When the invalid query is run, an **insert()**, **update()**, **deleteRecord()**, method on bad query results can result in data loss.

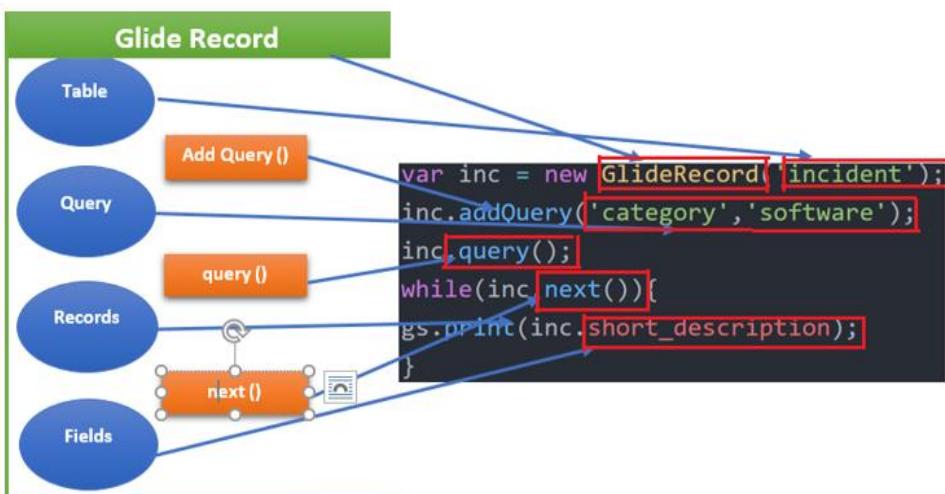
1. Most Common API
2. Running from server side
3. Used to generate SQL Queries
4. Perform CRUD operations

Glide Record Architecture

Instead of passing **SQL Queries**, we can use **Java Script**



Glide Record API Mapping



Glide Record Methods

Glide Record Methods	
Query()	Insert()
addQuery()	deleteRecord()
addActiveQuery()	update ()
addEncodedQuery()	initialize ()
addInactiveQuery()	deleteMultiple ()
next()	updateMultiple()
get()	addNullQuery()
setLimit()	addNotNullQuery()
orderBy()	autoSysFields()
orderByDesc()	canCreate()
getRowNumber()	canRead()
hasNext()	canWrite()
getRowCount()	canDelete()
chooseWindow()	changes()
addjoinQuery()	Find()
getClassDisplayValue()	getAttribute()
getDisplayValue()	getElement()
getLabel()	getFields()
getLink()	getValue()
getLocaton	has Attachments()
getRecordClassName()	insertWithReferences()
getRelatedLists()	isNewRecord()
getRowNumber()	isValid()
getTableName()	isValidField()
restoreLocation()	isValidRecord()
saveLocation()	newRecord()
setAbortAction()	setDisplayValue()
setLocation	setForceUpdate()
_next	setValue()
_query	setWorkflow()

Note: Need to use **Script-Background** application

Glide Record Exercises

1. How to get result(output) in Servicenow?

```
gs.print ('Welcome to Servicenow Academy');  
gs.info ('Welcome to Servicenow Academy');
```

```
gs.print('Welcome to Servicenow Academy');
```

Result → Welcome to Servicenow Academy

2. Write a simple program add two numbers

```
var a = 10;  
var b = 20;  
var c = a+b;  
gs.print (a+b);
```

```
1 var a = 10;  
2 var b = 20;  
3 var c = a+b;  
4 gs.print(a+b);
```

Result → 30

3. Working with query () method

```
var inc = new GlideRecord ('incident')  
//GlideRecord is main Object and Incident is Table  
inc.query(); //Query is execute in the table  
while (inc.next ()) { //Loop will runs on the table  
    gs.print (inc.number); //Printing all incidents  
}
```

```
1 var inc = new GlideRecord('incident')  
2 //GlideRecord is main Object and Incident is Table  
3 inc.query(); //Query is execute in the table  
4 while(inc.next()){ //Loop will runs on the table  
5     gs.print(inc.number); //Printing all incidents  
6 }
```

Result → Print all records numbers in Incident Table

4. Working with query() and addQuery() and next() and While methods

Exercise -1: Display priority -1 tickets from incident table with **addQuery** methods

```
var inc = new GlideRecord ('incident');
inc.addQuery ('priority=1'); // Add the query
inc.query ();
while(inc.next()){
gs.print(inc.number);
}
```

```
1  var inc = new GlideRecord('incident');
2  inc.addQuery('priority=1'); // Add the query
3  inc.query();
4 ~ while(inc.next()){
5    gs.print(inc.number);
6  }
```

Result → Printing all priority-1 tickets

5. Working with Multiple Queries

Exercise-2: Passing Multiple Queries using by same methods

```
var inc = new GlideRecord('incident');
inc.addQuery ('active', true);      //Query 1
inc.addQuery ('priority=1');        //Query 2
inc.addQuery ('category','software'); //Query 3
inc.query ();
while(inc.next()){
gs.print (inc.number);
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('active',true);          //Query 1
inc.addQuery('priority=1');           //Query 2
inc.addQuery('category','software'); //Query 3
inc.query();
while(inc.next()){
gs.print(inc.number);
}
```

Result → Print all records where your Condition meet

6. Working with addEncodedQuery () method

Exercise-3: we can use **addEncodedQuery** method
Instead of passing multiple queries into our script

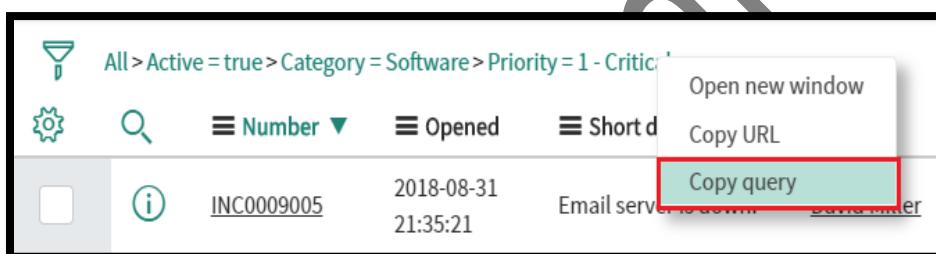
Step-1: Navigate to Incident **list view** and apply condition

Step-2: Condition: active = true and priority =1 and category = software

Step-3: Click on **Run**

Step-4: Copy applied query through **Copy query**

Step-5: Use this entire query into your script



Step-6: Script

```
var inc = new GlideRecord ('incident');
inc.addEncodedQuery('active=true^category=software^priority=1');
inc.query();
while(inc.next()){
gs.print(inc.number);
}
```

```
var inc = new GlideRecord('incident');
inc.addEncodedQuery('active=true^category=software^priority=1');
inc.query();
while(inc.next()){
gs.print(inc.number);
}
```

Exercise-4: Encoded Query set to a variable that variable to call into code

```
var ecq = 'active=true^category=software^priority=1';
//encodedquery set to a variable
var inc = new GlideRecord('incident');
inc.addEncodedQuery (ecq);
inc.query();
while (inc.next()){
gs.print (inc.number);
}
```

```
var ecq = 'active=true^category=software^priority=1';
//encodedquery set to a variable
var inc = new GlideRecord('incident');
inc.addEncodedQuery(ecq);
inc.query();
while(inc.next()){
gs.print(inc.number);
}
```

Result → Print all records where this meet
'active=true^category=software^priority=1';

7. Working with addQuery ('String','Operator','Value')

Practice with these all **SQL Operators** for better experience

- =
- !=
- >
- >=
- <
- <=

Strings (must be in upper case):

- =
- !=
- IN
- NOT IN
- STARTSWITH
- ENDSWITH
- CONTAINS

- DOES NOT CONTAIN
- INSTANCEOF

Exercise-5: Get Active and Priority is less than or equal to 2

```
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('priority','<=',2);
inc.query();
while(inc.next()){
  gs.print(inc.number);
}
```

```
1  var inc = new GlideRecord('incident');
2  inc.addActiveQuery();
3  inc.addQuery('priority','<=',2); //Using Operators
4  inc.query();
5  while(inc.next()){
6    gs.info(inc.number);
7 }
```

Result→ Print Critical-1 and High-2 tickets

Exercise-7: Working with SQL operators <= and CONTAINS

```
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('priority','<=',2);
inc.addQuery('short_description','CONTAINS','SAP');
inc.query();
while(inc.next()){
  gs.print(inc.number + ' ' + inc.short_description);
}
```

```

var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('priority','<=',2);
inc.addQuery('short_description','CONTAINS','SAP');
inc.query();
while(inc.next()){
    gs.print(inc.number + ' ' + inc.short_description);
}

```

Result→ Print all records where our condition meet like (≤ 2 and CONTAINS)

Exercise-8: Working with IN operator and print category of Software and Hardware

```

var cat = ['software', 'hardware'];
var inc = new GlideRecord('incident');
inc.addQuery('category', 'IN', cat);
inc.query();
while(inc.next()) {
    gs.print(inc.getValue('number') + ' ' + inc.getValue('short_description')) ;
}

```



```

var cat = ['software', 'hardware'];
var inc = new GlideRecord('incident');
inc.addQuery('category', 'IN', cat);
inc.query();
while(inc.next()) {
    gs.print(inc.getValue('number') + ' ' + inc.getValue('short_description'));
}

```

Result→ Print where category is Software and Hardware

Exercise-9: Working with STARTSWITH Operator

```

var inc = new GlideRecord('incident');
inc.addQuery('category', 'STARTSWITH', 'net');
inc.query();

```

```
while(inc.next()) {  
    gs.print(inc.number);  
}  
  
var inc = new GlideRecord('incident');  
inc.addQuery('category', 'STARTSWITH', 'net');  
inc.query();  
while(inc.next()) {  
    gs.print(inc.number);  
}
```

8. Working with addActiveQuery() method

Exercise-10: Instead of use **active=true** this method directly we can use **addActiveQuery**

```
var inc = new GlideRecord('incident');  
inc.addActiveQuery(); // instead if passing active = true  
inc.addQuery('priority',1);  
inc.query();  
while (inc.next ()) {  
    gs.info (inc.number);  
}  
  
1  var inc = new GlideRecord('incident');  
2  inc.addActiveQuery(); // instead if pass active = true  
3  inc.addQuery('priority',1);  
4  inc.query();  
5  while(inc.next()){  
6      gs.info(inc.number);  
7  }
```

Result→ Print all records where **active** is **true** and **priority-1**

9. Working with addInactiveQuery () method

Exercise-10: Instead of use **active=false** this method directly we can use **addInactiveQuery**

```
var inc = new GlideRecord ('incident');
inc.addInactiveQuery (); //Opposite of active query
inc.addQuery ('priority=1');
inc.query ();
while (inc.next ()) {
    gs.print (inc.number);
}
```

```
var inc = new GlideRecord('incident');
inc.addInactiveQuery(); //Opposite of active query
inc.addQuery('priority=1');
inc.query();
while(inc.next()){
    gs.print(inc.number);
}
//print only inactive records like--Closed
```

Result→ Print only inactive Records like Incident state is **Closed**

10. Working with getEncodedQuery () method

Exercise-11: getEncodedQuery from our code

```
var inc = new GlideRecord ('incident');
inc.addEncodedQuery ('active=true^category=software^priority=1');
inc.query();
while (inc.next ()) {
    gs.print (inc.getEncodedQuery ());
}
```

```
var inc = new GlideRecord ('incident');
inc.addEncodedQuery('active=true^category=software^priority=1');
inc.query();
while(inc.next()){
gs.print(inc.getEncodedQuery());
}
```

Example: 2

```
var inc = new GlideRecord ('incident');
inc.addEncodedQuery('active=true^category=software^priority=1');
inc.query();
gs.print(inc.getEncodedQuery());
```

Result → Print our encodedQuery

11. Working with orderBy() method

Exercise-12: Display all records in order wise
(Ascending) it depends on field values

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.addQuery('category=software');
inc.orderBy('short_description');
inc.query();
while(inc.next()){
gs.print(inc.number + ' ' + inc.short_description);
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.addQuery('category=software');
inc.orderBy('short_description');
inc.query();
while(inc.next()){
gs.print(inc.number + ' ' + inc.short_description);
}
```

Result→ Print all incidents order wise depends on **Short Description**

12. Working with orderByDesc () method

Exercise-12: Display all records in order wise
(Descending) it depends on field values

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.addQuery('category=software');
inc.orderByDesc('short_description');
inc.query();
while(inc.next()){
    gs.print(inc.number + ' ' + inc.short_description);
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.addQuery('category=software');
inc.orderByDesc('short_description');
inc.query();
while(inc.next()){
    gs.print(inc.number + ' ' + inc.short_description);
}
```

Result→ Print all records in descending order (**short_description**)

13. Working with setLimit () method

Exercise-13: Display limited records from specified table

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.orderByDesc('short_description');
```

```
inc.setLimit(10);
inc.query();
while(inc.next()){
gs.print(inc.number + ' ' + inc.short_description);
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.orderByDesc('short_description');
inc.setLimit(10);
inc.query();
while(inc.next()){
gs.print(inc.number + ' ' + inc.short_description);
}
```

Result→ Print only latest **10 records** created from given table

14. Working with get() Method

Exercise-14: Get record **sys_id** depends on **INC number** or Get incident record number depends on **sys_id**

```
var inc = new GlideRecord('incident');
inc.get('number','INC0009005');
gs.print(inc.sys_id);
```

```
var inc = new GlideRecord('incident');
inc.get('number','INC0009005');
gs.print(inc.sys_id);
```

Result→ Print sys_id related to incident number

Example-2

```
var inc = new GlideRecord('incident');
inc.get('sys_id','ed92e8d173d023002728660c4cf6a7bc');
gs.print(inc.number + ' ' + inc.short_description);
```

Result→ Print Incident number related to **sys_id**

15. Working with chooseWindow () method

Exercise-15: Display records between two numbers

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.addActiveQuery();
inc.chooseWindow(3,7)//include first value excluded secod value
inc.query()
while(inc.next()){
    gs.print(inc.number);
}
var inc = new GlideRecord('incident');
inc.addQuery('priority=1');
inc.addActiveQuery();
inc.chooseWindow(3,7)//include first value exclued secod value
inc.query()
while(inc.next()){
    gs.print(inc.number);
}
```

Result→ Print records from number 3 to 7 (4 records) **first number included and second number excluded**

16. Working with getRowCount () method

Exercise-16: Display all records from particular table (Incident)

```
var inc = new GlideRecord('incident');
inc.query()
gs.print(inc.getRowCount());
```

```
var inc = new GlideRecord('incident');
inc.query()
gs.print(inc.getRowCount()); //Display total number of records
```

Result→ Print number of records in particular table

Example-2: Display all active users in our **sys_user** tables

```
var inc = new GlideRecord('sys_user');
inc.addQuery('active=true');
inc.query();
gs.print ('Active users are:' + inc.getRowCount());
```

```
var inc = new GlideRecord('sys_user');
inc.addQuery('active=true');
inc.query();
gs.print('Active users are : ' + inc.getRowCount());
```

Result→ Print number of record in table

17. Working getTableName () method

Exercise-17: This method is used to get glide record table name

```
var inc = new GlideRecord ('change_request');
gs.print (inc.getTableName());
```

```
var inc = new GlideRecord('change_request');
gs.print(inc.getTableName());
```

Result→ Display current table name from glide record

18. Working getValue () method

Exercise-18: Get value of particular field in the table

```
var inc = new GlideRecord('incident');
inc.addQuery('active=true');
inc.query();
while(inc.next()){
    gs.print(inc.getValue('short_description'));
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('active=true');
inc.query();
while(inc.next()){
    gs.print(inc.getValue('short_description'));
}
```

Result→ Print the value of field from particular table

19. Working getDisplayValue () method

18. Print display value instead of actual value

```
var inc = new GlideRecord('incident');
inc.addQuery ('priority=1')
inc.query ();
while (inc.next ()){
    gs.print (inc.priority.getDisplayValue ());
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('priority=1')
inc.query();
while(inc.next()){
    gs.print(inc.priority.getDisplayValue());
}
```

Result→ Print display value of respective field

20. Working hasNext () method

Exercise-19: This method will return true if iterator have more elements.

```
var inc = new GlideRecord ('incident');
inc.query ();
gs.print (inc.hasNext ());
```

```
var inc = new GlideRecord('incident');
inc.query();
gs.print(inc.hasNext());
```

Result→ Print Boolean value (**True**)

21. Working with getUniqueValue () method

Exercise-20: Gets the unique key of the record, which is usually the sys_id unless otherwise specified.

```
var inc = new GlideRecord('incident');
inc.query();
inc.next();
var uniqvalue = inc.getUniqueValue();
gs.print(uniqvalue);
```

```
var inc = new GlideRecord('incident');
inc.query();
inc.next();
var uniqvalue = inc.getUniqueValue();
gs.print(uniqvalue);
```

22. Working with setValue () method

Exercise-22: This method is used to sets the value of the specific field with the specified value.

```
var attriName = 'category';
var inc = new GlideRecord ('incident');
inc.initialize ();
inc.setValue(attriName,'network');
inc.setValue('short_description','Critical VPN Issue');
inc.insert();
gs.print ('Category is ' + inc.category + ' and ' + 'issue is: ' +
inc.short_description);
```

```
var attriName = 'category';
var inc = new GlideRecord('incident');
inc.initialize();
inc.setValue(attriName, 'network');
inc.setValue('short_description','Critical VPN Issue');
inc.insert();
gs.print('Category is ' + inc.category + ' and ' + 'issue is: ' + inc.short_description);
```

Result→ Create a new record and **Set** a value into **category** field

23. Working with getElement () method

Exercise-23: This is used to get the specified column of the current record.

```
var elementName = 'short_description'
var inc = new GlideRecord('incident');
inc.initialize();
inc.setValue (elementName,'I am facing VPN Problem');
inc.insert ();
```

```
gs.print(inc.getElement('short_description'));

var elementName = 'short_description'
var inc = new GlideRecord('incident');
inc.initialize();
inc.setValue(elementName,'I am facing VPN Problem');
inc.insert();
gs.print(inc.getElement('short_description'));
```

Result→ Print current record column value

24. Working with getRecordClassName () method

Exercise-24: Retrieves the class name for the current record.

```
var inc = new GlideRecord('change_request');
var grcn = inc.getRecordClassName ();
gs.info (grcn);
```

```
var inc = new GlideRecord('change_request');
var grcn = inc.getRecordClassName ();
gs.info(grcn);
```

Result→ Print record class name (Table Name)

25. Working with initialize () and insert () method

Exercise-25: These methods are used to **Inserts a new record** using the field values that have been set for the current record

```
var inc = new GlideRecord ('incident');
inc.initialize (); //Compose incident form
inc.category = 'network'; // set field values
inc.short_description = 'Firewall Issue';
inc.priority = 1;
inc.insert (); // create new record
gs.print (inc.number);// print new record incident number
```

```
var inc = new GlideRecord('incident');
inc.initialize(); //Compose incident form
inc.category = 'network'; // set field values
inc.short_description = 'Firewall Issue';
inc.priority = 1;
inc.insert(); // create new record
gs.print(inc.number); // print new record incident number
```

Result → Create new record and print new record number

26. Working with isNewRecord () and newRecord () method

Exercise-26: Checks if the current record is a new record that has not yet been inserted into the database.

```
var inc = new GlideRecord ('incident');
inc.newRecord ();
gs.info (inc.isNewRecord());
```

```
var inc = new GlideRecord('incident');
inc.newRecord();
gs.info(inc.isNewRecord());
```

Result → Return boolean value true or false (**value is True**)

27. Working with isValid() method

Exercise-27: Define the current table exist or not. If table exist

display **true** not exist display **false**

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

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

Result → True

Example: 2

```
var inc = new GlideRecord ('srinivas');
gs.print (inc.isValid ());
```

```
var inc = new GlideRecord('srinivas');
gs.print(inc.isValid());
```

Result → False

28. Working with isValidField () method

Exercise-28: Determines if the specified field is defined in the current table. If Field exist in current record display true not exist display false

```
var inc = new GlideRecord ('incident');
gs.print (inc.isValidField ('category'));
```

```
var inc = new GlideRecord('incident');
gs.info(inc.isValidField('category'));
```

Result→ Boolean value is **True**

29. Working with getLink() and getProperty() method

Exercise-29: Retrieves a link to the current record.

```
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('category','software');
inc.addQuery('priority=1');
inc.query();
while(inc.next()){
    gs.print(gs.getProperty('glide.servlet.uri') + inc.getLink(false));
}
```

```
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('category','software');
inc.addQuery('priority=1');
inc.query();
while(inc.next()){
    gs.print(gs.getProperty('glide.servlet.uri') + inc.getLink(false));
}
```

Result→ Return the link of record

Example: 2 Return first record **Link** from query

```
var inc = new GlideRecord('incident');
inc.addActiveQuery();
inc.addQuery('category','software');
inc.addQuery('priority=1');
inc.query();
inc.next();
gs.print(gs.getProperty('glide.servlet.uri') + inc.getLink(false));
```

30. Working with isValidRecord () method

Exercise-30: Determines if a record was actually returned by the **query/get** record operation.

```
var inc = new GlideRecord ('incident');
inc.get ('number','INC0010012 ');
gs.print (inc.number + ' exists:' + inc.isValidRecord ());
```

```
var inc = new GlideRecord('incident');
inc.get('number','INC0010012 ');
gs.print(inc.number + ' exists : ' + inc.isValidRecord());
```

Result→ Display boolean value either true or false, **True**

31. Working with newRecord () method

Exercise-31: Creates a new GlideRecord record, sets the default values for the fields, and assigns a unique ID to the record.

```
var inc = new GlideRecord ('incident');
inc.newRecord ();
inc.short_description = 'Creating new record';
inc.category = 'software';
inc.insert ();
```

```
gs.print (inc.number);
```

```
var inc = new GlideRecord('incident');
inc.newRecord();
inc.short_description = 'Creating new record';
inc.category = 'software';
inc.insert();
gs.print(inc.number);
```

Result→ Create new record and print

32. Working with addNullQuery () method

Exercise-32: display all records where the value of the specified field is null.

```
var inc = new GlideRecord('incident');
inc.addNullQuery ('short_description')
inc.query ();
while (inc.next ()) {
gs.print (inc.number)
}
```

```
var inc = new GlideRecord('incident');
inc.addNullQuery('short_description')
inc.query();
while(inc.next()){
gs.print(inc.number)
}
```

Result→ Print all records where the specific field value is **Null**

33. Working with addNotNullQuery () method

Exercise-33: Opposite of addNullQuery methods display all records where the value of the specified field is **not null**.

```
var inc = new GlideRecord('incident');
inc.addNotNullQuery ('short_description')
inc.query ();
while (inc.next ()) {
gs.print (inc.number)
}
```

```
var inc = new GlideRecord('incident');
inc.addNotNullQuery ('short_description')
inc.query ();
while(inc.next()){
gs.print(inc.number)
}
```

Result→ Print all records where the specific field value is **not null**

34. Working with update () method single record

Exercise-34: Update specific record from table

```
var inc = new GlideRecord ('incident');
inc.get ('number','INC0000057');
inc.setValue ('state', 2);
inc.upd
ate ();
```

```
var inc = new GlideRecord('incident');
inc.get('number','INC0000057');
inc.setValue('state',2);
inc.update();
```

Result→ update record as expected

35. Working with updateMultiple () method multiple record

Exercise-35: Updates multiple records in a stated query with a specified set of changes from respected table.

```
var inc = new GlideRecord('incident');
inc.addQuery ('category', 'hardware');
inc.setValue('category', 'software');
inc.updateMultiple();
```

```
var inc = new GlideRecord('incident');
inc.addQuery('category', 'hardware');
inc.setValue('category', 'software');
inc.updateMultiple();
```

Result → Update multiple records as expected

Exercise: 2

```
var inc = new GlideRecord('incident');
inc.addQuery('active=true');
inc.query();
while (inc.next()) {
    inc.active = false;
    gs.print('Active incident ' + inc.number + ' closed');
    inc.update();
}
```

36. Working with **deleteRecord ()** method single record

Exercise-36: This method is used to delete single record from

```
var inc = new GlideRecord('incident');
inc.get ('number','INC0010013');// need to delete this record
inc.deleteRecord();
```

```
var inc = new GlideRecord('incident');
inc.get('number','INC0010013');// need to delete this record
inc.deleteRecord();
```

Result→ Delete single record as expected

37. Working with **deleteMultiple ()** method multiple record

Exercise-37: Deletes multiple records that satisfy the query condition.

```
var inc = new GlideRecord('incident');
inc.addQuery('priority', 4);
inc.query ();
inc.deleteMultiple();
```

```
var inc = new GlideRecord('incident');
inc.addQuery('priority',4);
inc.query();
inc.deleteMultiple();
```

Result→ Delete multiple records as expected

38. Working with canCreate () method

Exercise-38: Determines if the Access Control Rules, which include the user's roles, permit create new records in this table.

```
var inc = new GlideRecord ('incident');
gs.print (inc.canCreate());
```

```
var inc = new GlideRecord('incident');
gs.print(inc.canCreate());
```

Result→True (user have permission to create incident record)

39. Working with canRead () method

Exercise-39: Determines if the Access Control Rules, which include the user's roles, permit reading records in this table.

```
var inc = new GlideRecord('incident');
gs.print (inc.canRead());
```

```
var inc = new GlideRecord('incident');
gs.print(inc.canRead());
```

Result→True (user have permission to read incident record)

40. Working with canWrite () method

Exercise-40: Determines if the Access Control Rules, which include the user's roles, permit editing records in this table.

```
var inc = new GlideRecord ('incident');
gs.print (inc.canWrite ());
```

```
var inc = new GlideRecord('incident');
gs.print(inc.canWrite());
```

Result→True (user have permission to write incident record)

41. Working with canDelete () method

Exercise-41: Determines if the Access Control Rules, which include the user's roles, permit deleting records in this table.

```
var inc = new GlideRecord ('incident');
gs.print (inc.canDelete ());
```

```
var inc = new GlideRecord('incident');
gs.print(inc.canDelete());
```

Result→True (user have permission to delete incident record)

42. Working with autoSysFields () and setWorkflow () methods

Enables or disables the update to the fields, this is often used for manually updating field values on a record while leaving historical information unchanged.

sys_updated_by
sys_updated
sys_updated_on
sys_mod_count
sys_created_by
sys_created_on

These all system fields cannot update when we updated particular record via script

Note: autoSysFields method is not working on scoped application.

Note: setWorkflow (false) not run any other business rules

Exercise-43: Update multiple records without update any system fields

```
var inc = new GlideRecord ('incident');
inc.addQuery ('state', 1);
inc.query ();
while (inc.next ()) {
inc.autoSysFields (false);
inc.setWorkflow (false);
inc.setValue ('state', 2);
inc.update ();
}
```

```
var inc = new GlideRecord('incident');
inc.addQuery('state',1);
inc.query();
while(inc.next()){
    inc.autoSysFields(false);// Not update any system fields
    inc.setWorkflow(false);//Not running any business rule
    inc.setValue('state',2);
    inc.update();
}
```

Result → Updating records without update system fields

44. Working with addJoinQuery () methods

Exercise-44: Find problems that have an incident attached. This example returns problems that have associated incidents.

```
var prob = new GlideRecord('problem');
prob.addJoinQuery('incident', 'opened_by', 'caller_id');
prob.query();
while(prob.next()){
    gs.print(prob.number);
}
```

```

var prob = new GlideRecord('problem');
prob.addJoinQuery('incident', 'opened_by', 'caller_id');
prob.query();
while(prob.next()){
    gs.print(prob.number);
}

```

Result → Display all problem records associated incident

45. Working with getGlideObject () and getNumericValue () and setAbortAction methods

Exercise-45: This is method is used to cancel current action when condition is false

```

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

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

```

Lesson-02

Glide Form

Agenda

- Glide Form Overview and Usage
- Glide Form Methods
- Glide Form Exercises

Srinivas

Glide Form Overview and usage (g_form)?

- The **Glide Form** is client side API, mainly used to change default behavior of the current record
- GlideForm methods are only running on the client side (**Browser**).
- The global object **g_form** is used to access **GlideForm** methods.
- These methods are used to make custom changes to the form view of records. All validation of examples was done using Client Scripts.
- **g_form** methods frequently using in **Client Scripts**
- **g_form** methods can use in catalog client script also

Glide Form Methods

Glide Form Methods

setValue()	getFormElement()
getValue()	getSection()
addOption()	getTableName()
removeOption()	getUniqueValue()
addInfoMessage()	hideRelatedLists()
addErrorMessage()	isMandatory()
clearMessage()	save()
clearOptions()	submit()
disableAttachments()	setDisabled()
enableAttachments()	setDisplay()
Flash()	setMandatory()
getActionName()	setVisible()
getControl()	setReadOnly()
getElement()	setSectionDisplay()
showErrorBox()	showRelatedList()
showFieldMsg()	showRelatedLists()
addDecoration()	isNewRecord()
removeOption()	hideFieldMsg()

Practically work with these all methods

Glide Form Exercises

1. Navigate **Incident** table
2. Open one existing record
3. Click on **Ctrl+Shift+j**
4. Open **Java Script Executor**
5. Run our code here

Ctrl+Shift+j

1. Working with getValue () method

This method is used to get specific field value from current form.

Alert (g_form.getValue ('category')); //Alert is used to print the out put

```
alert(g_form.getValue('category'));
```

2. Working with setValue () method

Sets the value of specific field.

```
alert (g_form.getValue ('category'));
g_form.setValue ('category', 'hardware');
```

```
alert(g_form.getValue('category'));
g_form.setValue('category', 'hardware');
```

3. Working with setDisabled () method

This method is used to field make **Read only**

```
g_form.setDisabled ('category', true);
```

```
g_form.setDisabled ('category', true);
```

Note: UI Policy is best practice instead of using this method

4. Working with setMandatory () method

This method is used to a field make **Mandatory**

```
g_form.setMandatory ('category', true);
```

```
g_form.setMandatory ('category', true);
```

```
g_form.setMandatory ('category', false);
```

Note: UI Policy is best practice instead of using this method

5. Working with setDisplay () method

This method is used to **Hide** and **Un hide** specific fields

```
g_form.setDisplay ('business_service', false);
```

```
g_form.setDisplay ('business_service', false);
```

```
g_form.setDisplay ('business_service', true);
```

Note: UI Policy is best practice instead of using this method

6. Working with setVisible () method

This method is used to **Hide** field and maintain the space

```
g_form.setVisible ('subcategory', false);
```

```
g_form.setVisible('subcategory',false);
```

```
g_form.setVisible('subcategory',true);
```

7. Working with isMandatory () method

Defines whether the particular field is mandatory and must contain a value before the record can be saved or submitted display bullion (True/False)

```
g_form.isMandatory ('category');
```

```
g_form.isMandatory ('category');
```

8. Working with addInfoMessage() method

Add information message on top of the form

```
g_form.addInfoMessage ('please fill all mandatory fields');
```

```
g_form.addInfoMessage ('Please fill all mandatory fields');
```

9. Working with addErrorMessage() method

Add an error message on top of the form

```
g_form.addErrorMessage ('Fill mandatory fields');
```

```
g_form.addErrorMessage ('Fill mandatory fields');
```

10. Working with disableAttachments () method

Prevents the user attachment icon being hide

```
g_form.disableAttachments ();
```

```
g_form.disableAttachments();
```

11. Working with enableAttachments () method

Allows customer file attachments to be added. Shows the paper clip icon.

```
g_form.enableAttachments();
```

```
g_form.enableAttachments();
```

12. Working with clearMessages () method

Removes all **informational** and **error** messages from the top of the form.

```
g_form.addInfoMessage() and g_form.addErrorMessage().
```

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

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

13. Working with addOption () method

Add a new choice to respective field depends on requirement

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

```
g_form.addOption('impact', '4', '4 - Really Low');
```

14. Working with removeOption () method

a choice from respective field depends on requirement.

```
g_form.removeOption('impact', 4);
```

```
g_form.removeOption('impact', 4);
```

15. Working with clearOption () method

Removes all options from the choice list.

```
g_form.clearOptions('category');
```

```
g_form.clearOptions('category');
```

16. Working with clearValue () method

Remove value from specific field on the form

```
g_form.clearValue('category');
```

```
g_form.clearValue('category');
```

17. Working with hideRealtedLists () method

This method will hide all related lists on form

```
g_form.hideRelatedLists();
```

```
g_form.hideRelatedLists();
```

18. Working with showRealtedLists () method

This method will show all related lists on form

```
g_form.showRelatedLists();
```

```
g_form.showRelatedLists();
```

19. Working with isNewRecord () method

If record is new true, the record has never been saved.

```
g_form.isNewRecord();
```

```
g_form.isNewRecord();
```

20. Working with showFieldMsg () method

This method will display Informational or Error or Warn message under the specified form field (either a control object or the name of the field). If the control or field is off the screen, the form is scrolled to the field.

The **showErrorBox ()** method is an alternate method that does not require the type parameter.

```
g_form.showFieldMsg ('cmdb_ci','Atleast select one service app','info');
```

```
g_form.showFieldMsg('cmdb_ci','Atleast select one service app','info');
```

21. Working with hideFieldMsg () method

This method will hide the last message placed by **showFieldMsg ()**.

When true, all messages for the field are cleared. When false, only the last message is removed.

```
g_form.hideFieldMsg('impact');//Only the last message is removed.
```

```
g_form.hideFieldMsg('impact', true);//When true, all msgs for the field are cleared
```

```
g_form.hideFieldMsg('impact');//Only the last message is removed.
```

```
g_form.hideFieldMsg('impact', true);//When true, all messages for the field are cleared
```

```
g_form.showRelatedList('incident');//Pass related list table name
```

24. Working with getSections () and setSectionDisplay () methods

Display an array of the form's sections, which are available.

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

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

25. Working with getTableName () method

Returns the name of the table to which this record belongs.

```
alert (g_form.getTableName ());
```

```
alert(g_form.getTableName());
```

26. Working with getUniqueValue () method

It will return the sys_id of the record displayed in the form.

```
alert (g_form.getUniqueValue ());
```

```
    alert(g_form.getUniqueValue());
```

27. Working with addDecoration () method

Adds an icon on a field's label.

Adding the same item twice is prevented; however, you can add the same icon with a different title.

```
g_form.addDecoration ('caller_id', 'icon-star', 'Mark as Favorite', 'color-green');
```

```
g_form.addDecoration('caller_id', 'icon-star', 'Mark as Favorite', 'color-green');
```

28. Working with flash () method

This method is used to Flashes the specified color for a specified duration of time in the specified field.

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

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

Lesson-03

Glide User

Agenda

Glide User Overview and Usage

Glide User Methods

Glide User Exercises

Srinivas

Glide User Overview and Usage

- Glide User API and methods are useful to get **current logged** in user details and their roles. The typical use cases are personalizing feedback to the user and inspecting user roles.
- Glide user typically running from client side (**browser**)
- Very simple API in service now platform
- Typically used in **client scripts** and **UI policies** but is also found in **UI actions** that run on the client.
- Glide User Contains name and role information about the current user.
- The global object of Glide User is (**g_user**)
- This API contain different methods to use in our platform
- Determine if a user has a particular role assigned or not
- Using the **Glide User API** avoids the need to use the slower Glide Record queries to get user information.

Note: We cannot use in **Business Rules** or **UI Actions** that run on the server.

Glide User Methods

Glide User

firstName()	userID()
lastName()	hasRoles()
fullName()	userName()
hasRole()	getFullName()
hasRoleExacctly()	getClientData()
hasRoleFromList()	getClientName()

Practically work with these methods

Glide User Exercises

1. Navigate **Incident** table
2. Open one existing record
3. Click on **Ctrl+Shift+j**
4. Open **Java Script Executor**
5. Run our code here

Ctrl+Shift+J

1. Working with firstName () method

Display current user **First Name**

```
alert (g_user.firstName);
```

```
alert(g_user.firstName);
```

2. Working with lastName () method

Display current user **Last Name**

```
alert (g_user.lastName);
```

```
alert(g_user.lastName);
```

3. Working with fullName () method

Display current user **Full Name**

```
alert (g_user.fullName);
```

```
alert(g_user.fullName);
```

4. Working with userID () method

Display current 32-digit **user ID**

```
alert (g_user.userID);
```

```
alert(g_user.userID);
```

5. Working with hasRole () method

Display **true** if the current user has the **specified role**

```
alert(g_user.hasRole('admin'));
```

```
alert(g_user.hasRole('admin'));
```

6. Working with hasRoleExactly () method

Display **true** only if the current user has the specified role.

```
alert(g_user.hasRoleExactly('itil'));
```

```
alert(g_user.hasRoleExactly('itil'));
```

7. Working with hasRoleFromList() method

Returns **true** if the current user has at least one of the specified roles or has the admin role.

```
alert(g_user.hasRoleFromList('itil, admin'));
```

```
alert(g_user.hasRoleFromList('itil, admin'));
```

```
alert(g_user.hasRoles('itil, admin'));
```

8. Working with userName () method

Return current logged in user name

```
alert ('User Name is' + g_user.userName);
```

```
alert ('User Name is : ' + g_user.userName);
```

9. Working with fisrtName () and lastName () and userName () and userID () methods

This script will shows the difference between the **firstName**, **lastName**, **userName**, and **userID** property values.

```
alert("First Name = " + g_user.firstName  
+ ", \n Last Name = " + g_user.lastName  
+ ", \n User Name = " + g_user.userName  
+ ", \n User ID = " + g_user.userID);
```

```
alert("First Name = " + g_user.firstName  
+ ", \n Last Name = " + g_user.lastName  
+ ", \n User Name = " + g_user.userName  
+ ", \n User ID = " + g_user.userID);
```

Glide User from Scoped or Global

The GlideUser (**Scoped**) API provides access to information about the current user and current user roles. Using the GlideUser API avoids the need to use the slower GlideRecord queries to obtain user information.

Glide User Server or Scoped Methods

Glide User (Scoped or Global)

getCompanyID()	getID()
getDisplayNames()	getPreference()
getDomainID()	getRoles()
getFirstName()	getUserRoles()
getLastName()	isMemberOf()
getFullname()	hasRole()
getName()	savePreference()
getEmail()	getUserNames()

Practically work with these all methods

Note: Need to use Script-Background application

Glide User Scoped or Global Exercises

1. Navigate to **System Definition**
2. Open **Scripts – Background Application**
3. Write code into given space

1. Working with getFirstName() method

Returns the user's first name.

```
var cu = gs.getUser();
gs.print(cu.getFirstName());
```

```
var cu = gs.getUser();
gs.print(cu.getFirstName());
```

2. Working with getLastName() method

Returns the user's last name.

```
var cu = gs.getUser();
gs.print(cu.getLastName());
```

```
var cu = gs.getUser();
gs.print(cu.getLastName());
```

3. Working with getFullName() method

Returns the user's full name.

```
var cu = gs.getUser ();
gs.print (cu.getFullName ());
```

```
var cu = gs.getUser();
gs.print(cu.getFullName());
```

4. Working with getEmail() method

Returns the user's email address.

```
var cu = gs.getUser();  
gs.print(cu.getEmail());
```

```
var cu = gs.getUser();  
gs.print(cu.getEmail());
```

5. Working with getID() method

Returns the 32-digit sys_id of the current user.

```
var cu = gs.getUser();  
gs.print(cu.getID());
```

```
var cu = gs.getUser();  
gs.print(cu.getID());
```

6. Working with getName() method

Returns the user ID, or login name, of the current user.

```
var cu = gs.getUser();  
gs.print(cu.getName());
```

```
var cu = gs.getUser();  
gs.print(cu.getName());
```

7. Working with getCompanyID() method

Returns the current user's company sys_id.

```
var cu = gs.getUser();  
gs.print(cu.getCompanyID());
```

```
var cu = gs.getUser();  
gs.print(cu.getCompanyID());
```

8. Working with getDisplayName() method

Returns the current user's display name.

```
var cu = gs.getUser ();
gs.print (cu.getDisplayName());
```

```
var cu = gs.getUser();
gs.print(cu.getDisplayName());
```

9. Working with getDomainDisplayName() method

Returns the display value of the user's session domain.

```
var cu = gs.getUser ();
gs.print (cu.getDomainDisplayValue());
```

```
var cu = gs.getUser();
gs.print(cu..getDomainDisplayValue());
```

10. Working with getMyGroups() method

Returns an iterator containing the list of all groups to which the user belongs. Only active groups are returned.

```
var myGroupsArray = gs.getUser ().getMyGroups
().toArray ();
gs.print (myGroupsArray [0]);
```

```
var myGroupsArray = gs.getUser().getMyGroups().toArray();
gs.print(myGroupsArray[0]);
```

11. Working with getRoles() method

Returns a list of roles that includes explicitly granted roles, inherited roles, and roles acquired by group membership.

```
var cu = gs.getUser ();
gs.print (cu.getRoles());
```

```
var cu = gs.getUser();
gs.print(cu.getRoles());
```

12. Working with getUserByID () method

Returns the user object associated with the passed-in user ID (**sys_id** in **sys_user**) or **user_name**.

```
var newUser = gs.getUser();
gs.print(newUser.getUserByID ('abel.tuter').getFirstName());
```

```
var newUser = gs.getUser();
gs.print(newUser.getUserByID('abel.tuter').getFirstName());
```

13. Working with getUserRoles () method

Returns the list of roles explicitly granted to the user.

```
var cu = gs.getUser ();
gs.print(cu.getUserRoles());
```

```
var cu = gs.getUser();
gs.print(cu.getUserRoles());
```

14. Working with hasRole () method

Determines if the current user has the specified role.

```
var cu = gs.getUser ();
gs.print(cu.hasRole ('admin'));
```

```
var cu = gs.getUser();
gs.print(cu.hasRole('admin'));
```

15. Working with isMemberOf () method

Determines if the current user is a member of the specified group.

```
var cu = gs.getUser();
```

```
gs.print(cu.isMemberOf('database'));
```

```
var cu = gs.getUser();
gs.print(cu.isMemberOf('database'));
```

Srinivas Sunkara

Lesson-04

Glide System

Glide System

Glide System Overview

Glide System Methods

Glide System Exercises

Srinivas

Glide System Overview

- Glide System API running from server side
- This API will provide different methods to get information about the **system**, the current **Logged in user**, etc.
- The Glide System global object is **gs**
- Many of the Glide System methods facilitate the easy inclusion of dates in query ranges
- Often used in **filters** and **reporting**.

Glide System Methods

Glide System	
addInfoMessage()	endOfLastMonth()
addErrorMessage()	endOfLastWeek()
getUser()	endOfLastYear()
getUserName()	endOfNextMonth()
getUserID()	endOfNextWeek()
getUserDisplayName()	endOfNexYear()
hasRole()	endOfThisYear()
error()	endOfThisQuarter()
eventQueue()	endOfThisMonth
eventQueueScheduled()	hoursAgo()
getErrorMessage()	hoursAgoStart()
getSession()	minutesAgoEnd()
getSessionID()	minutesAgoStart()
setRedirect()	monthsAgo()
Info()	monthsAgoStart()
isDebugging()	nil()
isInteractive()	yearsAgo()
isLoggedIn()	yesterday()
getAvatar()	Warn()
setProperty()	tableExists()

Practically Work with these all methods

Note: Need to use Script-Background

Glide System Exercises

4. Navigate to **System Definition**
5. Open **Scripts – Background Application**
6. Write code into given space

1. Working with addInfoMessage () method

This method is used to add an info message for the current session.

This method is not supported for **asynchronous business rules**.

```
gs.addInfoMessage ('Start date should be always before end Date');
```

```
gs.addInfoMessage('Start date should be always before end Date');
```

Example: 2

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

2. Working with addErrorMessage () method

This method is used to add an error message for the current session.

```
gs.addErrorMessage ('please provide valid info');
```

```
gs.addErrorMessage('Please provide valid info');
```

3. Working with getUser () method

Returns a reference to the user object for the current user.

```
var currentUser = gs.getUser();  
gs.print(currentUser.getEmail());
```

```
var currentUser = gs.getUser();  
gs.print(currentUser.getEmail());
```

4. Working with getUserDisplayName () method

Return current user display name

```
gs.print(gs.getUserDisplayName());
```

```
gs.print(gs.getUserDisplayName());
```

5. Working with getUserID () method

Return current user **sys_id**

```
gs.print(gs.getUserID());
```

```
gs.print(gs.getUserID());
```

6. Working with getUserName () method

Display current user **user name**

```
gs.print(gs.getUserName());
```

```
gs.print(gs.getUserName());
```

7. Working with getUserNameByUserID () method

Returns the username based on a user ID.

```
gs.print(gs.getUserNameByUserID ('admin'));
```

```
gs.print(gs.getUserNameByUserID('admin'));
```

8. Working with hasRole () method

Define if the current user has at least one of the role assigned to current user

```
gs.print(gs.hasRole());
```

```
gs.print(gs.hasRole());
```

9. Working with hasRoleInGroup () method

Define if the current user has the specified role within a **specified group**.

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

```

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

```

10. Working with isInteractive() method

It will check if the current session is interactive or not

1. Interactive session is when a user logs in using the service now **log-in screen**
2. Non-interactive session is using a **SOAP request** to retrieve data from servicenw platform.

```
gs.print(gs.isInteractive());
```

```
gs.print(gs.isInteractive());
```

Example: 2

```

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

```

11. Working with getAvatar () method

Displays the file path to the current user's avatar.

```
var userPhoto = gs.getUser().getAvatar();  
gs.addInfoMessage ('User avatar ID: ' + userPhoto);
```

```
var userPhoto = gs.getUser().getAvatar();  
gs.addInfoMessage ('User avatar ID: ' + userPhoto);
```

12. Working with flushMessages () method

This method will clear all session messages saved using **addErrorMessage ()** or **addInfoMessage ()**.

Session messages are shown at the top of the form. In client side scripts, use **g_form.clearMessages ()** to remove session messages.

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

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

13. Working with beginningOfLastMonth () method

This method will get the date and time for the beginning of last month in GMT.

```
gs.print(gs.beginningOfLastMonth());
```

```
gs.print(gs.beginningOfLastMonth());
```

Output

```
*** Script: 2020-03-01 08:00:00
```

14. Working with **beginningOfLastWeek ()** method

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

```
gs.print(gs.beginningOfLastWeek());
```

Output

```
*** Script: 2020-04-06 07:00:00
```

15. Working with **beginningOfNextWeek ()** method

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

```
gs.print(gs.beginningOfNextWeek());
```

Output

```
*** Script: 2020-05-01 07:00:00
```

16. Working with **beginningOfNextYear ()** method

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

```
gs.print(gs.beginningOfNextYear());
```

Output

```
*** Script: 2021-01-01 08:00:00
```

17. Working with **beginningOfThisMonth ()** method

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

```
gs.print(gs.beginningOfThisMonth());
```

Output

```
*** Script: 2020-04-01 07:00:00
```

18. Working with **beginningOfThisQuarter ()** method

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

```
gs.print(gs.beginningOfThisQuarter());
```

Output

```
*** Script: 2020-04-01 07:00:00
```

19. Working with **beginningOfThisWeek ()** method

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

```
gs.print(gs.beginningOfThisWeek());
```

Output

```
*** Script: 2020-04-13 07:00:00
```

20. Working with **beginningOfThisYear ()** method

Returns the date and time for the beginning of this year in GMT.

```
gs.print(gs.beginningOfThisYear());
```

Output

```
*** Script: 2020-01-01 08:00:00
```

21. Working with beginningOfToday () method

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

```
gs.print(gs.beginningOfToday());
```

Output

```
*** Script: 2020-04-15 07:00:00
```

22. Working with beginningOfTomorrow () method

Retrieves the (**UTC**) beginning of tomorrow adjusted for the timezone of the current session

```
gs.print(gs.beginningOfTomorrow());
```

```
gs.print(gs.beginningOfTomorrow());
```

Output

```
*** Script: 2020-04-16 07:00:00
```

23. Working with beginningOfYesterday () method

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

```
gs.print(gs.beginningOfYesterday());
```

```
gs.print(gs.beginningOfYesterday());
```

Output

```
*** Script: 2020-04-14 07:00:00
```

24. Working with daysAgo () method

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

```
var gdt = new GlideDateTime()
```

```
gs.print(gs.daysAgo(4));
```

```
var gdt = new GlideDateTime()  
gs.print(gs.daysAgo(4));
```

25. Working with daysAgoEnd () method

Returns a date and time for the end of the day a specified number of days ago.

```
var gdt = new GlideDateTime()  
gs.print(gs.daysAgoEnd (10));
```

```
var gdt = new GlideDateTime()  
gs.print(gs.daysAgoEnd(10));
```

Output

```
*** Script: 2020-04-06 06:59:59
```

26. Working with daysAgoStart () method

Returns a date and time for the beginning of the day a specified number of days ago.

```
var gdt = new GlideDateTime ()  
gs.print (gs.daysAgoStart (4));
```

```
var gdt = new GlideDateTime()  
gs.print(gs.daysAgoStart(4));
```

Output

```
*** Script: 2020-04-11 07:00:00
```

27. Working with endOfLastMonth () method

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

```
gs.print(gs.endOfLastMonth());
```

```
gs.print(gs.endOfLastMonth());
```

Output

```
*** Script: 2020-04-01 06:59:59
```

28. Working with `endOfLastWeek()` method

Returns the date and time for the end of last week in GMT.

```
gs.print(gs.endOfLastWeek());
```

Output

```
*** Script: 2020-04-13 06:59:59
```

28. Working with `endOfLastYear()` method

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

```
gs.print(gs.endOfLastYear());
```

Output

```
*** Script: 2020-01-01 07:59:59
```

29. Working with `endOfNextMonth()` method

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

```
gs.print(gs.endOfNextMonth());
```

Output

```
*** Script: 2020-06-01 06:59:59
```

30. Working with `endOfNextWeek()` method

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

```
gs.print(gs.endOfNextWeek());
```

Output

```
*** Script: 2020-04-27 06:59:59
```

31. Working with `endOfNextYear()` method

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

```
gs.print(gs.endOfNextYear());
```

Output

```
*** Script: 2022-01-01 07:59:59
```

32. Working with `endOfThisMonth()` method

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

```
gs.print(gs.endOfThisMonth());
```

Output

```
*** Script: 2020-05-01 06:59:59
```

33. Working with `endOfThisQuarter()` method

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

```
gs.print(gs.endOfThisQuarter());
```

Output

```
*** Script: 2020-07-01 06:59:59
```

34. Working with `endOfThisWeek()` method

Returns the date and time for the end of this week in GMT.

```
gs.print(gs.endOfThisWeek());
```

Output

```
*** Script: 2020-04-20 06:59:59
```

35. Working with `endOfThisYear()` method

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

```
gs.print(gs.endOfThisYear());
```

Output

```
*** Script: 2021-01-01 07:59:59
```

36. Working with `endOfToday()` method

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

```
gs.print(gs.endOfToday());
```

Output

```
*** Script: 2020-04-17 06:59:59
```

37. Working with `endOfTomorrow()` method

Retrieves the date and time for the end of tomorrow in GMT.

```
gs.print(gs.endOfTomorrow());
```

Output

```
*** Script: 2020-04-18 06:59:59
```

38. Working with `endOfYesterday()` method

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

```
gs.print(gs.endOfYesterday());
```

Output

```
*** Script: 2020-04-16 06:59:59
```

39. Working with getDisplayColumn () method

Retrieves the display column for the table.

```
gs.print(gs.getDisplayColumn ('incident'));
```

```
gs.print(gs.getDisplayColumn ('incident'));
```

Output: number

40. Working with getErrorMessage () method

This method will get returns the list of error messages for the particular session that were added by **addErrorMessage ()**.

```
var errorMsg = gs.addErrorMessage ('Please fill mandatory fields');
gs.print (gs.getErrorMessages (errorMsg));
```

```
var errorMsg = gs.addErrorMessage ('Please fill mandatory fields');
gs.print(gs.getErrorMessages(errorMsg));
```

Output

Background message, type:error, message: Please fill mandatory fields
*** Script: []

41. Working with getInfoMessage () method

Retrieves the list of info messages for the session that were added by **addInfoMessage ()**.

```
var infoMsg = gs.addInfoMessage ('Info Message');
gs.print (gs.getInfoMessage (infoMsg));
```

```
var infoMsg = gs.addInfoMessage ('Info Message');
gs.print(gs.getInfoMessages(infoMsg));
```

Output

Background message, type:info, message: Info Message
*** Script: []

42. Working with getInitials () method

It will return the user's initials

```
var userInitials = gs.getUser ().getInitials();
gs.addInfoMessage ('User initials: ' + userInitials);
```

```
var userInitials = gs.getUser().getInitials();
gs.addInfoMessage('User initials: ' + userInitials);
```

OutPut

Background message, type:info, message: User initials: SA

43. Working with getMessage () method

Retrieves translated messages to display in the User Interface.

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

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

```
var myMsg = gs.getMessage ('This is my own message');
gs.print (myMsg);
```

```
var myMsg = gs.getMessage('This is my own message');
gs.print(myMsg);
```

Output

*** Script: This is my own message

44. Working with getSession () method

Returns a GlideSession object.

```
gs.print (gs.getSession ());
```

```
gs.print(gs.getSession());
```

Output

```
*** Script: com.glide.sys.GlideSnapshotSession@13f5bef
```

45. Working with getSessionID () method

Returns the GlideSession Session ID.

```
gs.print(gs.getSessionID());
```

```
gs.print(gs.getSessionID());
```

Output

```
*** Script: 55D56F9C079C101094BBFC289C1ED017
```

46. Working with hoursAgo() method

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

```
gs.print(gs.hoursAgo(7));
```

```
gs.print(gs.hoursAgo(7));
```

Output

```
*** Script: 2020-04-16 22:31:49
```

47. Working with hoursAgoStart () method

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

```
gs.print(gs.hoursAgoStart(2));
```

```
gs.print(gs.hoursAgoStart(2));
```

Output

```
*** Script: 2020-04-17 03:00:00
```

48. Working with isFirstDayOfMonth () method

This method will check whether the date is the first day of the month.

```
gs.print(gs.isFirstDayOfMonth(2020-01-25));
```

Output

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

49. Working with isFirstDayOfWeek () method

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

```
gs.print(gs.isFirstDayOfWeek(2020-04-12));
```

Output

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

50. Working with isFirstDayOfYear () method

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

```
gs.print(gs.isFirstDayOfYear(2020-04-12));
```

Output

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

51. Working with isLastDayOfMonth () method

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

```
gs.print(gs.isLastDayOfMonth(2020-04-19));
```

```
gs.print(gs.isLastDayOfMonth(2020-04-19));
```

Output

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

52. Working with isLastDayofWeek () method

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

```
gs.print(gs.isLastDayOfWeek (2020-04-19));
```

```
gs.print(gs.isLastDayOfWeek(2020-04-19));
```

Output

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

53. Working with isLastDayofYear () method

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

```
gs.print(gs.isLastDayOfYear (2020-04-19));
```

```
gs.print(gs.isLastDayOfYear(2020-04-19));
```

Output

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

54. Working with lastWeek () method

Returns the date and time one week ago in GMT.

```
gs.print(gs.lastWeek());
```

```
gs.print(gs.lastWeek());
```

Output

```
*** Script: 2020-04-10 10:11:38
```

55. Working with now () method

Returns the current date in UTC.

```
gs.print(gs.now());
```

```
gs.print(gs.now());
```

Output

```
*** Script: 2020-04-17
```

56. Working with nowDateTime () method

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

```
gs.print(gs.nowDateTime());
```

Output

```
*** Script: 2020-04-17 03:26:15
```

57. Working with nowDateTime () method

Returns the current date and time in UTC format.

```
gs.print(gs.nowNoTZ());
```

Output

```
*** Script: 2020-04-17 10:30:12
```

58. Working with setProperty () method

Sets the specified key to the specified value.

```
gs.setProperty("glide.foo","bar","foo");
gs.info(gs.getProperty("glide.foo"));
```

```
gs.setProperty("glide.foo","bar","foo");
gs.info(gs.getProperty("glide.foo"));
```

59. Working with setRedirect () method

Sets the redirect URI for this transaction, which then determines the next page the user will see.

```
gs.setRedirect("com.glideapp.servicecatalog_cat_item_vie
w.do?sysparm_id=d41ce5bac611227a0167f4bf8109bf70&
sysparm_user=" + current.sys_id + "&sysparm_email=" +
current.email);
```

Lesson-05

Glide Session

Glide Session

Glide Session Overview

Glide Session Methods

Glide Session Exercise

Srinivas

Glide Session Overview and Usage

The Glide Session allows us to find information about the current session
Glide Session API provides a way to find information on current session.

Glide Session Methods

Glide Session	
clearClientData()	isLoggedIn()
getClientData()	putClientData()
getLanguage()	getTimeZoneName
getRoles()	isInteractive()
Work with these all methods	

Note: Need to use Script-Background

Glide Session Exercises

1. Navigate to **System Definition**
2. Open **Scripts – Background** Application
3. Write code into given space

1. Working with getClientData () method

Returns a session client value previously set with **putClientData ()**.

This method is used in a client script to retrieve data values that were set by a server script that used the **putClientData ()** method.

Code

```
var myData = gs.getSession ();
myData.putClientData('userName','Srinivas');
var clientData = session.getClientData('userName')
gs.print (clientData);
```

```
var myData = gs.getSession();
myData.putClientData('userName', 'Srinivas');
var clientData = session.getClientData('userName');
gs.print(clientData);
```

Output

```
*** Script: Srinivas
```

2. Working with clearClientData () method

Clears a session client value previously set with **putClientData()**.

This method is used in a client script to clear data values that were set by a server script using the **putClientData ()** method.

Code

```
var myData = gs.getSession ();
myData.putClientData('userName', 'Srinivas');
var clientData = session.getClientData ('userName');
gs.print(clientData);
```

```
session.clearClientData ('userName');
clientData = session.getClientData('userName');
gs.print (clientData);
```

```
var myData = gs.getSession();
myData.putClientData('userName', 'Srinivas');
var clientData = session.getClientData('userName');
gs.print(clientData);

session.clearClientData('userName');
clientData = session.getClientData('userName');
gs.print(clientData);
```

Output

```
*** Script: Srinivas
*** Script: null
```

3. Working with getLanguage () method

This method is used to gets the session's language code.

Code

```
var session = gs.getSession();  
var language = session.getLanguage();  
gs.info(language);
```

```
var session = gs.getSession();  
var language = session.getLanguage();  
gs.info(language);
```

Output

```
*** Script: en
```

4. Working with getRoles () method

This method is used to gets a list of roles for the current logged in user.

The list of roles does not reflect any changes made during the current user session. To get the updated list of roles, the user must log out and log back in.

```
gs.info(gs.getSession().getRoles());
```

```
gs.info(gs.getSession().getRoles());
```

Output

```
*** Script: admin,sn_templated_snip.template_snippet_admin,sn_templated_snip.template_snippet_reader,
```

5. Working with getTimeZoneName() method

This method is used to gets the name of the session's time zone.

Code

```
var session= gs.getSession ();
var timeZone = session.getTimeZoneName();
gs.print (timeZone);
```

```
var session= gs.getSession();
var timeZone = session.getTimeZoneName();
gs.print(timeZone);
```

Output

```
*** Script: US/Pacific
```

6. Working with isInteractive () method

Determines if the current session is interactive.

Code

```
var interActive = gs.getSession().isInteractive();
gs.info (interActive);
```

```
var interActive = gs.getSession().isInteractive();
gs.info(interActive);
```

Output

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

7. Working with isInteractive () method

Determines if the current user is currently logged in.

Code

```
var session = gs.getSession ();
var loggedIn = session. IsLoggedIn ();
gs.info (loggedIn);
```

```
var session = gs.getSession();
var loggedIn = session.isLoggedIn();
gs.info(loggedIn);
```

Output

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

8. Working with putClientData () method

Sets a session client value that can be retrieved with **getClientData ()**. This method is used in a server side script that runs when a form is created.

Code

```
var session = gs.getSession();
var clientData = gs.putClientData('userName','Srinivas')
gs.print (clientData);
```

```
var session = gs.getSession();
var clientData = gs.putClientData('userName','Srinivas')
gs.print(clientData);
```

Lesson-06

Glide Date

Glide Date

Glide Date and Usage

Glide Date Methods

Glide Date Exercises

Srinivas

Glide Date Overview and Usage

Glide Date classes will provide methods for performing operations on Glide Date objects, such as instantiating Glide Date objects or working with Glide Date fields.

- These methods are used to work with date fields on form
- Glide Date global object is **GlideDate**

Glide Date Methods

Glide Date	
glideDate()	getMonthNoTZ()
getByFormat()	getValue()
getDayOfMonthNoTZ()	getYearNoTZ()
getDisplayValue()	setDisplayValue()
getDispalyValueInternal()	setValue()
	Subtract()

Practice with these all methods

Note: Need to use **Script-Background** application

Glide Date Exercises

8. Navigate to **System Definition**
9. Open **Scripts – Background Application**
10. Write code into given space

1. Working with GlideDate () method

Creates a GlideDate object with the current date time.

Code

```
var date = new GlideDate();
gs.print (date);
```

```
var date = new GlideDate();
gs.print(date);
```

2. Working with getByFormat () method

This method is used to gets the date in the specified date format.

Code

```
var date = new GlideDate ();
gs.print (date.getByFormat ('dd-MM-yyyy'));
```

```
var date = new GlideDate();
gs.print(date.getByFormat('dd-MM-yyyy'));
```

Output

```
*** Script: 18-04-2020
```

3. Working with getDayOfMonthNoTZ() method

Gets the day of the month stored by the GlideDate object, expressed in the UTC time zone.

```
var glideDate =new GlideDate();
gs.print (glideDate.getDayOfMonthNoTZ());
```

```
var glideDate =new GlideDate();
gs.print(glideDate.getDayOfMonthNoTZ());
```

Output

```
*** Script: 18
```

4. Working with getDayOfMonthNoTZ() method

This method is used to gets the date in the current logged in user's display format and time zone.

```
var glideDate =new GlideDate ();
gs.print (glideDate.getDisplayValue());
```

```
var glideDate =new GlideDate();
gs.print(glideDate.getDisplayValue());
```

Output

*** Script: 2020-04-18

5. Working with getDisplayValueInternal() method

Gets the display value in the internal format (yyyy-MM-dd).

Code

```
var glideDate =new GlideDate();
gs.print (glideDate.getDisplayValueInternal());
```

```
var glideDate =new GlideDate();
gs.print(glideDate.getDisplayValueInternal());
```

Output

*** Script: 2020-04-18

6. Working with getMonthNoTZ() method

Gets the month stored by the GlideDate object, expressed in the UTC time zone.

Code

```
var glideDate =new GlideDate();
gs.print (glideDate.getMonthNoTZ());
```

```
var glideDate =new GlideDate();
gs.print(glideDate.getMonthNoTZ());
```

Output

*** Script: 4

7. Working with getValue () method

Gets the date value stored in the database by the GlideDate object in the internal format, **yyyy-MM-dd**, and the system time zone, **UTC by default**.

Code

```
var glideDate =new GlideDate();
gs.print (glideDate.getValue());
```

```
var glideDate =new GlideDate();
gs.print(glideDate.getValue());
```

Output

```
*** Script: 2020-04-18
```

8. Working with getYearNoTZ () method

Gets the year stored by the GlideDate object, expressed in the UTC time zone.

Code

```
var glideDate =new GlideDate();
gs.print (glideDate.getYearNoTZ());
```

```
var glideDate =new GlideDate();
gs.print(glideDate.getYearNoTZ());
```

Output

```
*** Script: 2020
```

9. Working with setDisplayValue () method

Sets a date value using the current user's display format and time zone.

Code

```
var glideDate =new GlideDate();
glideDate.setDiaplayValue();
gs.print (glideDate.getValue());
```

```
var glideDate =new GlideDate();
glideDate.setDiaplayValue('18-04-2020');
gs.print(glideDate.getValue());
```

10. Working with setValue () method

Sets the date of the GlideDate object.

Code

```
var glideDate =new GlideDate();
glideDate.setValue('2020-04-18');
gs.print (glideDate.getValue());
```

```
var glideDate =new GlideDate();
glideDate.setValue('2020-04-18');
gs.print(glideDate.getValue());
```

Output

*** Script: 2020-04-18

11. Working with subtract() method

Gets the duration difference between two GlideDate values.

Code

```
var firstDate = new GlideDate();
firstDate.setDisplayValue('2020-01-25');
var secondDate = new GlideDate();
secondDate.setDisplayValue('2020-01-29');

difference= GlideDate.subtract(firstDate, secondDate);
gs.info(difference.getDisplayValue());
```

```
var firstDate = new GlideDate();
firstDate.setDisplayValue('2020-01-25');
var secondDate = new GlideDate();
secondDate.setDisplayValue('2020-01-29');

difference= GlideDate.subtract(firstDate, secondDate);
gs.info(difference.getDisplayValue());
```

Output

```
*** Script: 4 Days
```

Lesson-07

Glide Date and Time

Agenda

GlideDateTime Overview and usage

GlideDateTime Methods

Glide Date Time Exercises

Srinivas

GlideDateTime Overview and usage?

The GlideDateTime class provides methods for performing operations on GlideDateTime objects, such as instantiating GlideDateTime objects or working with **glide_date_time** fields.

- GlideDateTime object, performing date-time calculations,
- Formatting a date-time, or converting between date-time formats.
- GlideDateTime running from server side
- Global object is **GlideDateTime**

GlideDateTime Methods

Glide Date Time	
add()	getDate()
GlideDateTime()	getDayOfMonth()
addDays()	getdayOfMonthLocalTime()
addDaysLocalTime()	getDayOfMonthUTC()
addDaysUTC()	getDayOfWeek()
addMonths()	getDayOfWeekLocalTime()
addMonthsUTC()	getDayOfMonthUTC()
addMonthsLocalTime()	getDayOfWeek
addSeconds()	getDaysInMonth()
addWeeks()	getDaysInMonthLocalTime()
addWeeksLocalTime()	getDaysInMonthUTC()
addWeeksUTC()	getDisplayValue()
addYears()	getDisplayValueInteranal()
addYearsLocalTime()	getDSTOffset()
compareTo()	getErorMsg()
equals()	getLocalDate()
getNumericValue()	getLocalTime()
getTime()	getMonth()
getTZOffeset()	getMonthLocalTime()
getUserTimeZone()	getMonthUTC()
getUTCMidnight()	isValid()
getValue()	setDayOfMonth()
getWeekOfYearLocalTime()	setDayOfMonthLocalTime()

getWeekOfYearUTC()	setDayOfMonthUTC()
getYear()	setDisplayValue()
getYearLocalTime()	setDisplay
getYearUTC()	setGlideDateTime()
Subtrac()	toString()

Note: Need to use **Script-Background** application

Glide Date Time Exercises

1. Navigate to **System Definition**
2. Open **Scripts – Background** Application
3. Write code into given space

1. Working with GlideDateTime () method

Instantiates a new **GlideDateTime** object with the current date and time in GMT format.

```
var gdt = new GlideDateTime ();
gs.print (gdt);
```

```
var gdt = new GlideDateTime();
gs.print(gdt);
```

Output

```
*** Script: 2020-04-19 09:44:28
```

2. Working with addDays () method

This method is used to add a specified number of days to the current **GlideDateTime** object. A negative parameter subtracts days.

Code

```
var gdt = new GlideDateTime ();
gdt.addDays (4)
gs.print (gdt);
```

```
var gdt = new GlideDateTime();
gdt.addDays(4)
gs.print(gdt);
```

Output

*** Script: 2020-04-23 14:00:28

3. Working with addDaysLocalTime () method

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days.

The method determines the local date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts days using the local date and time values.

Code

```
var gdt = new GlideDateTime("2020-01-28 08:00:00");
gdt.addDaysLocalTime(4)
gs.print(gdt.getLocalDate());
```

```
var gdt = new GlideDateTime("2020-01-28 08:00:00");
gdt.addDaysLocalTime(4)
gs.print(gdt.getLocalDate());
```

Output

*** Script: 2020-02-01

4. Working with addDaysUTC () method

Adds a specified number of days to the current GlideDateTime object. A negative parameter subtracts days.

The method determines the UTC date and time equivalent to the value stored by the GlideDateTime object, then adds or subtracts days using the UTC date and time values.

Code

```
var gdt = new GlideDateTime("2020-01-28 08:00:00");
gdt.addDaysUTC(4)
gs.print(gdt.getDate());
```

```
var gdt = new GlideDateTime("2020-01-28 08:00:00");
gdt.addDaysUTC(4)
gs.print(gdt.getDate());
```

Output

```
*** Script: 2020-02-01
```

5. Working with addMonths () method

This method is used to adds a specified number of months to the current GlideDateTime object. A negative parameter subtracts months.

Code

```
var gdt = new GlideDateTime();
gdt.addMonths(2)
gs.print(gdt.getDate());
```

```
var gdt = new GlideDateTime();
gdt.addMonths(2)
gs.print(gdt.getDate());
```

Output

```
*** Script: 2020-06-19
```

6.Similarly working with

addMonthsLocalTime () and

addMonthsUTC () methods

addMonthsLocalTime ()

```
var gdt = new GlideDateTime();
gdt.addMonthsLocalTime(2)
gs.print(gdt.getDate());
```

Output

*** Script: 2020-06-19

addMonthsUTC ()

```
var gdt = new GlideDateTime();
gdt.addMonthsUTC(2)
gs.print(gdt.getDate());
```

Output

*** Script: 2020-06-19

7. Working with addSeconds() method

Adds a specified number of seconds to the GlideDateTime object.

Code

```
var gdt = new GlideDateTime("2018-04-20 08:00:00");
gdt.addSeconds(45);
gs.print(gdt.getValue());
```

```
var gdt = new GlideDateTime("2018-04-20 08:00:00");
gdt.addSeconds(45);
gs.print(gdt.getValue());
```

Output

*** Script: 2018-04-20 08:00:45

8. Working with addWeeks () method

Adds a specified number of weeks to the current GlideDateTime object. A negative parameter subtracts weeks.

Use addWeeksLocalTime () and addWeeksUTC () instead of this method.

Code

```
var gdt = new GlideDateTime();
gdt.addWeeks(5);
gs.print(gdt.getValue());
```

```
var gdt = new GlideDateTime();
gdt.addWeeks(5);
gs.print(gdt.getValue());
```

Output

*** Script: 2020-05-24 14:43:28

9. Similarly working with addWeeksLocalTime () and addWeeksUTC () methods addWeeksLocalTime ()

Code

```
var gdt = new GlideDateTime();
gdt.addWeeksLocalTime(5);
gs.print(gdt.getValue());
```

```
var gdt = new GlideDateTime();
gdt.addWeeksLocalTime(5);
gs.print(gdt.getValue());
```

Output

*** Script: 2020-05-24 14:47:17

addWeeksUTC ()

```
var gdt = new GlideDateTime();
gdt.addWeeksUTC(5);
gs.print(gdt.getValue());
```

Output

```
*** Script: 2020-05-24 14:50:30
```

```
*** Script: 0  
*** Script: 1  
*** Script: -1
```

Adds a specified number of years to the current GlideDateTime object. A negative parameter subtracts years.

Use **addYearsLocalTime ()** or **addYearsUTC ()** instead of this method.

Code

```
var gdt = new GlideDateTime();  
gdt.addYears(3);  
gs.print(gdt.getValue());
```

```
var gdt = new GlideDateTime();  
gdt.addYears(3);  
gs.print(gdt.getValue());
```

Output

```
*** Script: 2023-04-19 14:57:37
```

11. Similarly working with [addYearsLocalTime \(\)](#) and [addYearsUTC \(\)](#) methods

[addYearsLocalTime \(\)](#)

```
var gdt = new GlideDateTime();  
gdt.addYearsLocalTime(3);  
gs.print(gdt.getValue());
```

Output

```
*** Script: 2023-04-19 15:05:58
```

[addYearsUTC \(\)](#)

```
var gdt = new GlideDateTime();
gdt.addYearsUTC(3);
gs.print(gdt.getValue());
```

Output

*** Script: 2023-04-19 15:08:21

12. Working with compareTo () method

Compares two date and time objects to determine whether one occurs before the other or if they are equivalent.

Code

```
var initDate = new GlideDateTime ("2011-08-01 12:00:00");
var compDate1 = new GlideDateTime("2011-08-01 12:00:00");
var compDate2 = new GlideDateTime("2011-07-31 12:00:00");
var compDate3 = new GlideDateTime("2011-08-04 16:00:00");

gs.info (initDate.compareTo(compDate1)); // Equals (0)
gs.info (initDate.compareTo(compDate2)); // initDate is after
compDate2 (1)
gs.info (initDate.compareTo(compDate3)); // initD
```

```
var initDate = new GlideDateTime("2011-08-01 12:00:00");
var compDate1 = new GlideDateTime("2011-08-01 12:00:00");
var compDate2 = new GlideDateTime("2011-07-31 12:00:00");
var compDate3 = new GlideDateTime("2011-08-04 16:00:00");

gs.info(initDate.compareTo(compDate1)); // Equals (0)
gs.info(initDate.compareTo(compDate2)); // initDate is after compDate2 (1)
gs.info(initDate.compareTo(compDate3)); // initD
```

Output

*** Script: 0
*** Script: 1
*** Script: -1

13. Working with equals () method

Compares an object with an existing value for equality.

Code

```
var gdt = new GlideDateTime("2020-04-19 00:00:00");
gs.print(gdt.equals("2020-04-15 00:00:00"));
```

```
var gdt = new GlideDateTime("2020-04-19 00:00:00");
gs.print(gdt.equals("2020-04-15 00:00:00"));
```

Output

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

14. Working with getDate () method

Gets the date stored by the GlideDateTime object, expressed in the standard format, **yyyy-MM-dd**, and the system time zone, UTC by default.

Code

```
var gdt = new GlideDateTime();
gs.print(gdt.getDate());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDate());
```

Output

```
*** Script: 2020-04-19
```

15. Working with getDayOfMonth () method

Gets the current day of the month in the UTC time zone.

Use **getDayOfMonthLocalTime ()** and **getDayOfMonthUTC ()** instead of this method.

Code

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfMonth());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfMonth());
```

Output

```
*** Script: 19
```

16. Similarly working with getDayOfMonthLocalTime () and getDayOfMonthUTC () method

Code

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfMonthLocalTime());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfMonthLocalTime());
```

Output

```
*** Script: 19
```

getDaysOfMonthUTC ()

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfMonthUTC());
```

Output

```
*** Script: 20
```

17. Working with getDayOfWeek () method

Retrieves the day of the week stored by the GlideDateTime object, expressed in the user's time zone.

Use **getDayOfWeekLocalTime ()** and **getDayOfWeekUTC ()** instead of this method.

Code

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfWeek());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfWeek());
```

Output

```
*** Script: 7
```

18. Similarly working with getDayOfWeekLocalTime () and getDayOfWeekUTC () method

getDayOfWeekLocalTime ()

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfWeekLocalTime());
```

Output

```
*** Script: 1
```

getDayOfWeekUTC ()

```
var gdt = new GlideDateTime();
gs.print(gdt.getDayOfWeekUTC());
```

Output

```
*** Script: 1
```

19. Working with getDaysInMonth () method

Gets the number of days in the month stored by the GlideDateTime object, expressed in the Java Virtual Machine time zone.

Use `getDaysInMonthLocalTime()` and `getDaysInMonthUTC()` instead of this method.

```
var gdt = new GlideDateTime();
gs.print(gdt.getDaysInMonth());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDaysInMonth());
```

*** Script: 30

20. Similarly working with `getDaysInMonthsLocalTime()` and `getDayInMonthsUTC()` method

`getDaysInMonthsLocalTime()`

```
var gdt = new GlideDateTime();
gs.print(gdt.getDaysInMonthLocalTime());
```

Output

*** Script: 30

`getDayInMonthsUTC()`

```
var gdt = new GlideDateTime();
gs.print(gdt.getDaysInMonthUTC());
```

Output

*** Script: 30

21. Working with `getDisplayValue()` method

Gets the date and time value in the current user's display format and time zone

```
var gdt = new GlideDateTime();
gs.print(gdt.getDisplayValue());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDisplayValue());
```

Output

```
*** Script: 2020-04-20 00:53:17
```

22. Working with getDisplayValueInternal () method

Returns the display value in the internal format (yyyy-MM-dd HH:mm:ss). This method is useful for date/time fields, but not for date fields.

```
var gdt = new GlideDateTime();
gs.print (gdt.getDisplayValueInternal ());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDisplayValueInternal());
```

Output

```
*** Script: 2020-04-20 01:03:24
```

23. Working with getDisplayValueInternal () method

Gets the amount of time that daylight saving time is offset.

```
var gdt = new GlideDateTime();
gs.print (gdt.getDSTOffset ());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getDSTOffset());
```

Output

```
*** Script: 3600000
```

24. Working with getErrorMsg () method

This method is used to gets the current error message.

```
var gdt = new GlideDateTime("2011-08-31 aa:00:00"); //bad  
gs.print(gdt.isValid()); //false  
gs.print (gdt.getErrorMsg()); //reason; //reason
```

```
var gdt = new GlideDateTime("2011-08-31 aa:00:00"); //bad  
gs.print(gdt.isValid()); //false  
gs.print(gdt.getErrorMsg()); //reason; //reason
```

Output

```
*** Script: true  
*** Script: null
```

25. Working with getLocalDate () method

Gets the date stored by the GlideDateTime object, expressed in the standard format, **yyyy-MM-dd**, and the current user's time zone.

```
var gdt = new GlideDateTime();  
gs.print (gdt.getLocalDate());
```

```
var gdt = new GlideDateTime();  
gs.print(gdt.getLocalDate());
```

Output

```
*** Script: 2020-04-20
```

26. Working with getLocalTime () method

Gets the time in the user's time zone.

```
var gdt = new GlideDateTime('2020-04-20 09:11:10');  
gs.print (gdt.getLocalTime());
```

```
var gdt = new GlideDateTime('2020-04-20 09:11:10');
gs.print(gdt.getLocalTime());
```

Output

```
*** Script: 1970-01-01 02:08:15
```

27. Working with getMonth () method

Retrieves the month stored by the GlideDateTime object, expressed in Java Virtual Machine time zone.

Use **getMonthLocalTime ()** and **getMonthUTC ()** instead of this method.

```
var gdt = new GlideDateTime();
gs.print (gdt.getMonth());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getMonth());
```

Output

```
*** Script: 4
```

28. Working with getNumericValue () method

Gets the number of milliseconds since January

```
var gdt = new GlideDateTime();
gs.print (gdt.getNumericValue());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getNumericValue());
```

Output

```
*** Script: 1587375814685
```

29. Working with getValue () method

Gets the date and time value stored by the GlideDateTime object in the internal format, yyyy-MM-dd HH:mm:ss, and the system time zone, UTC by default.

```
var gdt = new GlideDateTime();
gs.print (gdt.getValue());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getValue());
```

Output

```
*** Script: 2020-04-20 10:00:37
```

30. Working with getYear () method

Retrieves the year stored by the GlideDateTime object, expressed in the Java Virtual Machine time zone.

Use **getYearLocalTime()** and **getYearUTC()** instead of this method.

```
var gdt = new GlideDateTime();
gs.print (gdt.getYear());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.getYear());
```

Output

```
*** Script: 2020
```

31. Similarly working with **getYearLocalTime()** and **getYearUTC()** method

getYearLocalTime()

```
var gdt = new GlideDateTime();
gs.print(gdt.getYearLocalTime());
```

Output

*** Script: 2020

getYearUTC()

```
var gdt = new GlideDateTime();
gs.print(gdt.getYearUTC());
```

Output

*** Script: 2020

32. Working with **hasDate ()** method

Determines if an object's date is set.

```
var gdt = new GlideDateTime();
gs.print(gdt.hasDate());
```

```
var gdt = new GlideDateTime();
gs.print(gdt.hasDate());
```

Output

*** Script: true

33. Working with **setDayOfMonth ()** method

Sets the day of the month to a specified value.

Use **setDayOfMonthLocalTime(day)** and **setDayOfMonthUTC(day)** instead of this method.

```
var gdt = new GlideDateTime();
gdt.setDayOfMonth(4);
gs.print(gdt.getDayOfMonth());
```

```
var gdt = new GlideDateTime();
gdt.setDayOfMonth(4);
gs.print(gdt.getDayOfMonth());
```

Output

```
*** Script: 4
```

34. Similarly working with **setDayOfMonthLocalTime ()** and **setDayOfMonthUTC ()** method

SetDayOfMonthLocalTime ()

```
var gdt = new GlideDateTime();
gdt.setDayOfMonthLocalTime(4);
gs.print(gdt.getDayOfMonth());
```

Output

```
*** Script: 4
```

setDayOfMonthUTC ()

```
var gdt = new GlideDateTime();
gdt.setDayOfMonthUTC(4);
gs.print(gdt.getDayOfMonth());
```

Output

```
*** Script: 3
```

35. Working with **setDisplayValue ()** method

Sets a date and time value using the current user's display format and time zone.

```
var gdt = new GlideDateTime("2011-02-02 12:00:00");
gdt.setDisplayValue("2011-01-01 12:00:00");
gs.print(gdt.getValue());
```

```
var gdt = new GlideDateTime("2011-02-02 12:00:00");
gdt.setDisplayValue("2011-01-01 12:00:00");
gs.print(gdt.getValue());
```

Output

```
*** Script: 2011-01-01 20:00:00
```

36. Working with **setMonth ()** method

Sets the month stored by the GlideDateTime object to a specified value using the Java Virtual Machine time zone.

Use **setMonthLocalTime ()** or **setMonthUTC ()** instead of this method.

```
var gdt = new GlideDateTime();
gdt.setMonth(1);
gs.print(gdt.getMonth());
```

```
var gdt = new GlideDateTime();
gdt.setMonth(1);
gs.print(gdt.getMonth());
```

Output

```
*** Script: 1
```

37. Similarly working with [setMonthLocalTime \(\)](#) and [setMonthUTC \(\)](#) method

setMonthLocalTime ()

```
var gdt = new GlideDateTime();
gdt.setMonthLocalTime(1);
gs.print(gdt.getMonth());
```

Output

```
*** Script: 1
```

setMonthUTC ()

```
var gdt = new GlideDateTime();
gdt.setMonthUTC(1);
gs.print(gdt.getMonth());
```

Output

```
*** Script: 1
```

38. Working with [setValue \(\)](#) method

Sets the date and time of the GlideDateTime object.

```
var gdt = new GlideDateTime("2011-01-01 12:00:00");
gdt.setValue ("2011-02-02 08:00:00");
gs.print(gdt.getValue());
```

```
var gdt = new GlideDateTime("2011-01-01 12:00:00");
gdt.setValue ("2011-02-02 08:00:00");
gs.print(gdt.getValue());
```

Output

```
*** Script: 2011-02-02 08:00:00
```

39. Working with setYear () method

Sets the year stored by the GlideDateTime object to a specified value using the UTC time zone.

```
var gdt = new GlideDateTime();
gdt.setYearUTC(2020);
gs.print(gdt.getYearUTC());
```

```
var gdt = new GlideDateTime();
gdt.setYearUTC(2020);
gs.print(gdt.getYearUTC());
```

Output

```
*** Script: 2020
```

40. Working with subtract () method

Subtracts a specified amount of time.

```
var gdt = new GlideDateTime("2011-08-31 08:00:00");
var gtime1 = new GlideTime();
gtime1.setValue("00:00:20");
gdt.subtract(gtime1);
gs.print(gdt.getTime());
```

```
var gdt = new GlideDateTime("2011-08-31 08:00:00");
var gtime1 = new GlideTime();
gtime1.setValue("00:00:20");
gdt.subtract(gtime1);
gs.print(gdt.getTime());
```

Output

```
*** Script: 1970-01-01 07:59:40
```

41. Working with toString () method

Returns the date and time value stored by the GlideDateTime object in the internal format, yyyy-MM-dd HH:mm:ss, and the system time zone, UTC by default.

This method is equivalent to **getValue ()**.

```
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.toString());
```

```
var gdt = new GlideDateTime("2011-08-31 08:00:00");
gs.print(gdt.toString());
```

Output

```
*** Script: 2011-08-31 08:00:00
```

Lesson-08

Glide Aggregation

Agenda

Glide Aggregate Overview

Glide Aggregate Methods

Glide Aggregate Exercises

Srinivas

Glide Aggregate Overview

GlideAggregate enables you to easily create database aggregation queries.

The scoped GlideAggregate class is an extension of Glide Record and provides database aggregation (**COUNT, SUM, MIN, MAX, AVG**) queries. This functionality can be helpful when creating customized reports or in calculations for calculated fields. The GlideAggregate class works only on number fields.

The following shows how to add aggregates to a query on the category and software fields in the Incident [incident] table.

Exercise-1: Getting count of records which are in active and category is software

```
var count = new GlideAggregate('incident');
count.addEncodedQuery('active=true');
count.addQuery('category=software');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next())
    incidents = count.getAggregate('COUNT');
gs.info(incidents);
```

```
var count = new GlideAggregate('incident');
count.addEncodedQuery('active=true');
count.addQuery('category=software');
count.addAggregate('COUNT');
count.query();
var incidents = 0;
if (count.next())
    incidents = count.getAggregate('COUNT');
gs.info(incidents);
```

Exercise-2: The following code shows how to add aggregates to a query on the category and software fields in the Incident [incident] table.

```
var incidentGA = new GlideAggregate('incident');
incidentGA.addQuery('category', 'software');
incidentGA.setGroup(false);
incidentGA.addAggregate('COUNT', 'sys_mod_count');
incidentGA.addAggregate('SUM', 'sys_mod_count');
incidentGA.addAggregate('AVG', 'sys_mod_count');
incidentGA.addAggregate('MIN', 'sys_mod_count');
incidentGA.addAggregate('MAX', 'sys_mod_count');
incidentGA.addAggregate('STDDEV', 'sys_mod_count');
incidentGA.query();
if (incidentGA.next()) {
    gs.info('COUNT: ' + incidentGA.getAggregate('COUNT',
'sys_mod_count'));
    gs.info('SUM: ' + incidentGA.getAggregate('SUM', 'sys_mod_count'));
    gs.info('AVG: ' + incidentGA.getAggregate('AVG', 'sys_mod_count'));
    gs.info('MIN: ' + incidentGA.getAggregate('MIN', 'sys_mod_count'));
    gs.info('MAX: ' + incidentGA.getAggregate('MAX', 'sys_mod_count'));
    gs.info('STDDEV: ' + incidentGA.getAggregate('STDDEV',
'sys_mod_count'));
}
```

```

var incidentGA = new GlideAggregate('incident');

incidentGA.addQuery('category', 'software');

incidentGA.setGroup(false);

incidentGA.addAggregate('COUNT', 'sys_mod_count');
incidentGA.addAggregate('SUM', 'sys_mod_count');
incidentGA.addAggregate('AVG', 'sys_mod_count');
incidentGA.addAggregate('MIN', 'sys_mod_count');
incidentGA.addAggregate('MAX', 'sys_mod_count');
incidentGA.addAggregate('STDDEV', 'sys_mod_count');

incidentGA.query();

if (incidentGA.next()) {
    gs.info('COUNT: ' + incidentGA.getAggregate('COUNT', 'sys_mod_count'));
    gs.info('SUM: ' + incidentGA.getAggregate('SUM', 'sys_mod_count'));
    gs.info('AVG: ' + incidentGA.getAggregate('AVG', 'sys_mod_count'));
    gs.info('MIN: ' + incidentGA.getAggregate('MIN', 'sys_mod_count'));
    gs.info('MAX: ' + incidentGA.getAggregate('MAX', 'sys_mod_count'));
    gs.info('STDDEV: ' + incidentGA.getAggregate('STDDEV', 'sys_mod_count'));
}

```

Exercise-3: Number of incidents varies depending on the current state of the incident table

~~Simplifies~~

```

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

```

```

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

```

Result:

There are currently 4 incidents with a category of
 There are currently 1 incidents with a category of database
 There are currently 5 incidents with a category of hardware
 There are currently 18 incidents with a category of inquiry
 There are currently 4 incidents with a category of network
 There are currently 8 incidents with a category of software

Exercise-4: Count all incidents opened each quarter in a year

```

var ga = new GlideAggregate('incident');
ga.addAggregate('COUNT'); // Count all incidents opened each quarter
ga.addTrend('opened_at', 'quarter');
ga.query();
while(ga.next()) {
    gs.info([ga.getValue('timeref'), ga.getAggregate('COUNT')]);
}

```

```

var ga = new GlideAggregate('incident');
ga.addAggregate('COUNT'); // Count all incidents opened each quarter
ga.addTrend('opened_at', 'quarter');
ga.query();
while(ga.next()) {
    gs.info([ga.getValue('timeref'), ga.getAggregate('COUNT')]);
}

```

Exercise-5: Count all incidents with each category

```
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
    gs.info(count.getAggregateEncodedQuery()); }
```

```
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
    gs.info(count.getAggregateEncodedQuery());
}
```

Result

```
Script: category=
Script: category=database
Script: category=hardware
Script: category=inquiry
Script: category=network
Script: category=software
```

Exercise-6: Count all incidents with each category

Retrieves the number of rows in the table with aggregation

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

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

Result

```
6
Update counts: MIN = 8 MAX = 12 AVG = 9.5000
Database Update counts: MIN = 8 MAX = 50 AVG = 29.0000
Hardware Update counts: MIN = 2 MAX = 15 AVG = 6.8000
Inquiry / Help Update counts: MIN = 0 MAX = 36 AVG = 8.3636
Network Update counts: MIN = 3 MAX = 37 AVG = 18.0000
Software Update counts: MIN = 4 MAX = 95 AVG = 21.0769
```

```
var agg = new GlideAggregate('incident');
agg.addAggregate('AVG', 'sys_mod_count');
agg.groupBy('category');
agg.query();
while (agg.hasNext()) {
    agg.next();
    var avg = agg.getAggregate('AVG', 'sys_mod_count');
    var category = agg.category.getDisplayValue();
    gs.info(category + ': AVG = ' + avg);
}
```

Lesson-09

Client Scripts

Agenda

- Client Script Overview**
- Types of Client Scripts**
- Why client scripts**
- Client side Glide API classes**
- Client Script Exercises**

Srinivas

Client Scripts OverView

Client Scripts are allowing to user to run Java Script on client side (**Browser Window**) when client based events are occurred or executed

Such as when a form was, **Loads and or submitted form** and when a **field value is changed** and when you **edit cell value from list view**

We can be able to use client scripts to configure **Forms, Form Fields and Field Values** while user is using any particular form.

Client Scripts can do some below activities like

1. Make it as field **Mandatory or Optional**
2. It is used to display message based on **Field Values**
3. Make is as field **Read only and Writable**
4. Make it as filed **Visible and Invisible**
5. Add and modify the options in a choice list based on **user's role**
6. Giving **Info and Error Messages** to customer
7. **Add and Remove** the choice values
8. **Set and Get** values from fields
9. Display **Flash** messages
10. Add **Information Message** to customers

Types of Client Scripts

There are 4 types of client scripts in Service Now such as

onLoad ()

OnLoad client script will runs when the system first loaded on the form and before users can enter data. Typically, onLoad () client scripts mainly running from client-side of the current form and t is used to set default record values.

onChange ()

on Change will runs when a particular field value changes on the form. then we can use onchange client script

onSubmit ()

it will run when a form is submitted. Typically, onSubmit () scripts validate things on the form and ensure that the submission makes sense. An onSubmit () client script can cancel form submission by returning a value of false.

OnCellEdit ()

It will run when a user tries to change a cell value from list view then onCellEdit client script will be execute and provide result

Why Client Scripts?

Client scripts can be reducing the amount of time it takes while a user to complete a form.

If we create client scripts is used controlling a field values on a form, you must use another method to control these field values in a list.

Using by client scripts we can perform following operations:

1. We can restrict a user List Editing for the particular table.
2. Will create particular Business rules or we can access controls for list editing.
3. We can create Data policies and UI Policy
4. On list view we can restrict a user onCellEdit client script.
5. Check validation when we submit form
6. Validation will check when form was loaded
7. It can be reduced the need for time-consuming from server lookups.
8. **g_scratchpad** and asynchronous **GlideAjax** lookup are simple way get the information from the server Glide Record and **g_form. getReference ()** methods are also available for retrieving server Information.
9. **However**, these methods are no longer recommended due to their performance impact. So always we can recommend to use **g_scratchpad** and **asynchronous GlideAjax**

Use of g_scratchpad

The **g_scratchpad** object will pass the information from the Server to the Client, when we want to get required information that is not available on the particular form.

Where Our Client Scripts Run

With the exception of **onCellEdit () client scripts**, remaining client scripts only apply on particular forms and search pages, **onLoad ()**, **onChange ()**, **onSubmit ()**

If you want create a client script to controlling field values on a **form**, you should be use one of these other methods to control field values when on a list.

1. We can create an access control rules to restrict user who can read and edit field values on your form. Create a business rules to validate content in form.
2. Create a UI Policy and Data policy to validate content in form.
3. Create an onCellEdit () client script to validate content in list view.
4. Disable list editing for the particular table

Client side Glide API classes

GlideAjax : This object is used to execute server-side script from the client

GlideForm : Customize our forms

GlideList2 : Used to customize (v2) lists, including normal lists and related lists.

GlideUser : This object is used to get session information about the current user and roles.

GlideMenu : It is used to customize UI Context Menu items.

GlideDialogWindow : Use this class to display a dialog window

Client Scripts Exercises

Excercise: 1

Requirement

When **Incident** Form Loaded then some filed are make **Mandatory** or **Read Only** or **Visible /Invisible** and add an **Information Message** to define fill all mandatory fields

1. Number and Priority Fields are **Read only**
2. Caller and Short Description fields are **Mandatory**
3. Configuration Item field is should not be seen (**Invisible**)
4. Display an **Information Message** top on the form if the record is new

Client Script Type

OnLoad

Procedure

1. Navigate to **System Definition**→**Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

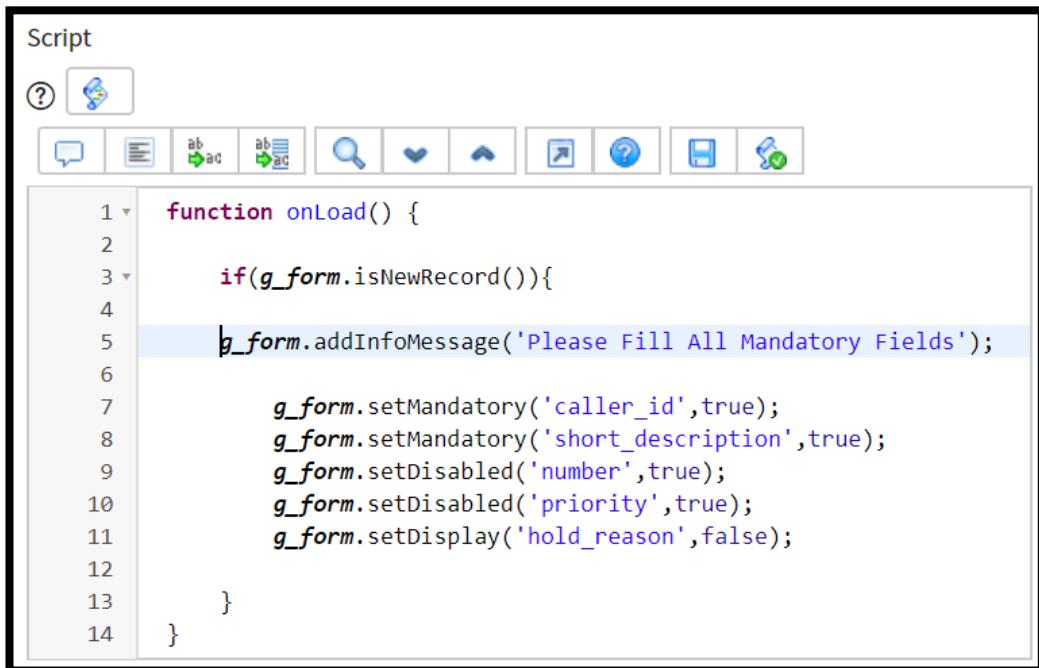
The screenshot shows the Service Catalog interface with the 'Client Scripts' list. The 'New' button is highlighted with a red box. The list includes entries for 'Maximum Duration Range Check' and 'Set protocol based on auth type'.

4. Fill Clients Scripts Form
5. **Name:** Initial Form Validation
6. **Table:** Incident
7. **UI Type:** All
8. **Type:** OnLoad

The screenshot shows the 'Client Script' configuration form for 'Initial Form Validation'. The 'Table' field is set to 'Incident [incident]' and the 'Type' field is set to 'onLoad', both highlighted with red boxes. The 'Description' field contains a detailed explanation of the script's purpose and functionality.

Name	Initial Form Validation	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onLoad	Global	<input checked="" type="checkbox"/>
Description	When Incident Form Loaded then some filed are make Mandatory and Read only and Invisible and also add an Information Message to define fill all mandatory fields 1. Number and Priority Fields are Read only 2. Caller and Short Description fields are Manadatory 3. On-Hold reason field should not be seen (Invisible) 4. Display an Information Message top on the form if the record is new		

9. Script

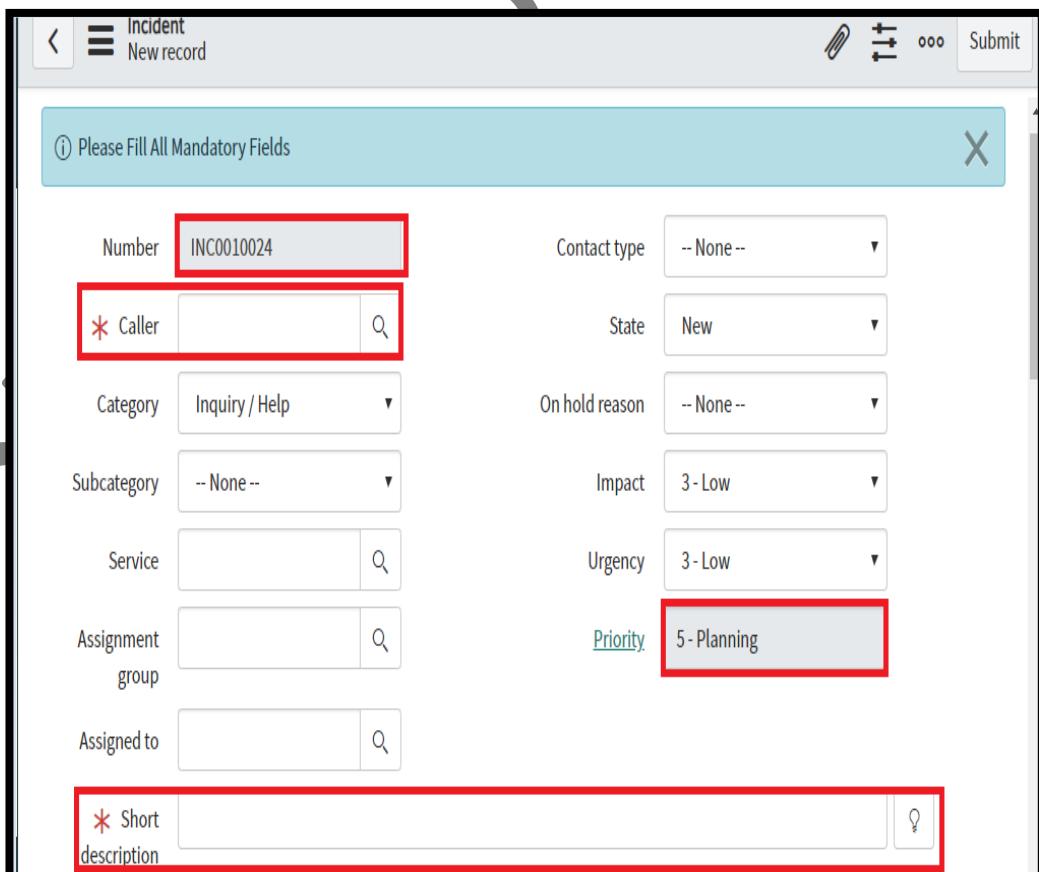


```
function onLoad() {
    if(g_formisNewRecord()){
        g_form.addInfoMessage('Please Fill All Mandatory Fields');

        g_form.setMandatory('caller_id',true);
        g_form.setMandatory('short_description',true);
        g_form.setDisabled('number',true);
        g_form.setDisabled('priority',true);
        g_form.setDisplay('hold_reason',false);
    }
}
```

10. Open New Incident form check the validation

11. Incident → Click on Create New



The screenshot shows the 'New record' screen for an 'Incident' form. A modal dialog box at the top displays the message: 'Please Fill All Mandatory Fields'. Below the dialog, several input fields are highlighted with red boxes to indicate they are mandatory:

- Number: INC0010024
- * Caller: (empty)
- Category: Inquiry / Help
- Subcategory: -- None --
- Service: (empty)
- Assignment group: (empty)
- Assigned to: (empty)
- * Short description: (empty)

Other fields shown but not highlighted include Contact type, State, On hold reason, Impact, Urgency, and Priority (set to 5 - Planning).

Exercise: 2

Requirement

1. When Incident State is **On-Hold** Then On Hold Reason field should be **Visible** and **Mandatory**
2. Add warning message below hold reason field that is "Please select at least one hold reason"

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

The screenshot shows the 'New' dialog for a client script. The 'Name' field is 'State On-Hold Hold reason', 'Application' is 'Global', and the 'Active' checkbox is checked. The 'Table' dropdown is set to 'Incident [incident]' (highlighted with a red box). The 'UI Type' dropdown is 'All'. The 'Type' dropdown is set to 'onChange' (highlighted with a red box). The 'Field name' dropdown is set to 'State' (highlighted with a red box). In the 'Description' section, there is a requirement note:

Requirement:

1. When Incident State is On-Hold Then On Hold Reason field should be Visible and Mandatory
2. Add warning message below hold reason field that is "Please select at least one hold reason"

4.Script

Script

```

1 function onChange(control, oldValue, newValue, isLoading,
2   isTemplate) {
3     if (isLoading || newValue === '') {
4       return;
5     }
6     if(newValue == 3){
7       g_form.setMandatory('hold_reason',true);
8       g_form.setDisplay('hold_reason',true);
9       g_form.showFieldMsg('hold_reason','Please select
10      atleast one hold reason','warning',false);
11    }else{
12      g_form.setMandatory('hold_reason',false);
13      g_form.setDisplay('hold_reason',false);
14    }
15  }

```

5. Open **New Incident** form check the validation
6. **Incident → Click on Create New**
7. Change state value to **On-Hold**
8. Check on hold reason field is **Visible** and **Mandatory**
9. Follow below screen shot for reference

Incident
New record

Number	INC0010057	Contact type	-- None --
* Caller	<input type="text"/>	State	On Hold
Category	Inquiry / Help	* On hold reason	-- None --
Subcategory	-- None --	Please select atleast one hold reason	
Service	<input type="text"/>	Impact	3 - Low
Configuration	<input type="text"/>		

Exercise: 3

Requirement:

Any customer after submitted (**Priority 1-Critical**) incident

1. We need to display one alert message to customer about **Criticality** (Are you submitting Priority one ticket)
2. Include **Ok** and **Cancel** Buttons also

Client Script Type

OnSubmit

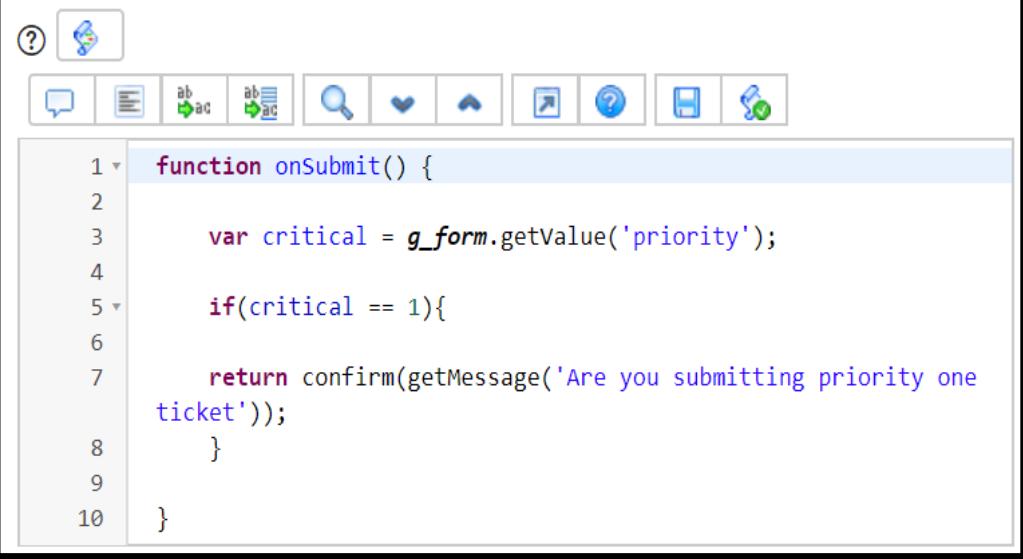
Procedure

1. Navigate to **System Definition→Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

Name	Warnig Msg about criticality	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onSubmit	Global	<input checked="" type="checkbox"/>
Description	Requirement: Any customer after submitted (Priority 1-Critical) incident 1. We need to display one alert message to customer about Criticality 2. Include Ok and Cancel Buttons also		

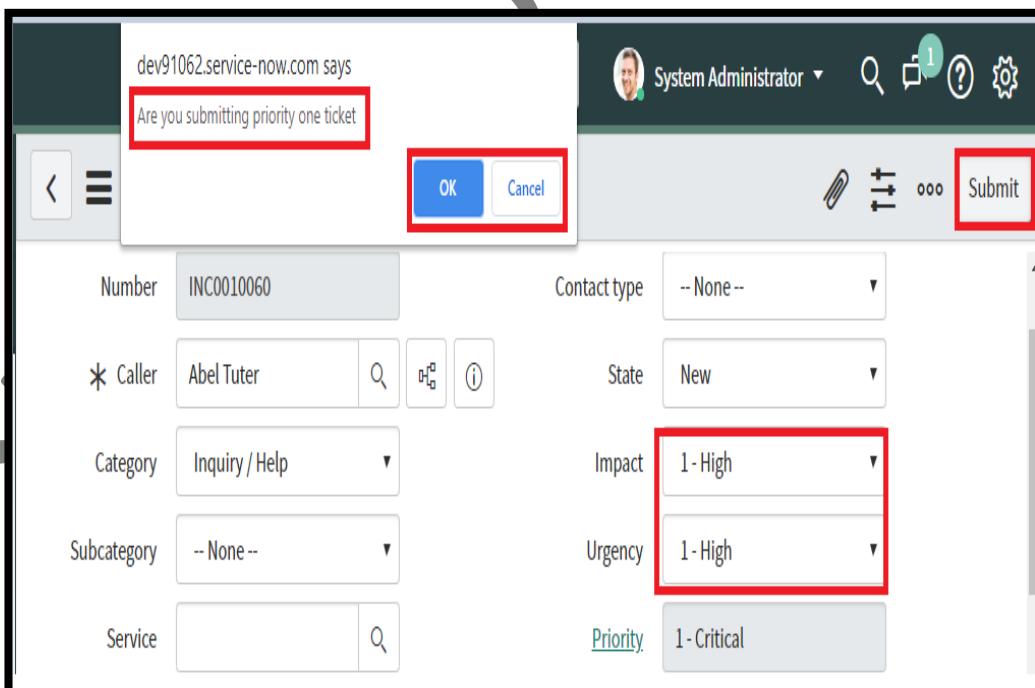
4. Script

Script



```
function onSubmitted() {  
    var critical = g_form.getValue('priority');  
  
    if(critical == 1){  
        return confirm(getMessage('Are you submitting priority one ticket'));  
    }  
}
```

5. Open **New Incident** form check the validation
6. **Incident** → Click on **Create New**
7. Select **Impact-1** and **Urgency-1**
8. Click on **Submit** Button
9. Follow below screen shot for reference



The screenshot shows the 'New Incident' form in ServiceNow. A modal dialog box is open, asking 'Are you submitting priority one ticket?' with 'OK' and 'Cancel' buttons. The 'Submit' button on the form is also highlighted with a red box. The form fields are as follows:

Field	Value
Number	INC0010060
Contact type	-- None --
* Caller	Abel Tuter
State	New
Category	Inquiry / Help
Impact	1 - High
Subcategory	- None --
Urgency	1 - High
Service	
Priority	1 - Critical

Exercise: 4

Requirement

1. When Incident state is **Resolved** Then **Resolution Code** and **Resolution Notes** are mandatory
2. **Assigned to** and **Assignment Group** should not be seen

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition**→**Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

The screenshot shows the 'New Client Script' dialog box. The fields are as follows:

Name	State Resolved Make Validati	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	State		
Description	Requirement 1. When Incident state is Resolved Then Resolution Code and Reolution Notes are mandatory 2. Assigned to and Assignment Group should not be seen		

4. Script

Script

```

1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2    if (isLoading || newValue === '') {
3      return;
4    }
5
6    if(newValue == 6){
7
8      g_form.setMandatory('close_code',true);
9      g_form.setMandatory('close_notes',true);
10     g_form.setDisplay('assigned_to',false);
11     g_form.setDisplay('assignment_group',false);
12   }else {
13
14     g_form.setMandatory('close_code',false);
15     g_form.setMandatory('close_notes',false);
16     g_form.setDisplay('assigned_to',true);
17     g_form.setDisplay('assignment_group',true);
18   }
19
20 }

```

5. Open New Incident form check the validation
6. **Incident** → Click on **Create New**
7. Change state value to **Resolved**
8. Check validation on below screen shot for reference

Exercise: 5

Requirement

1. If Selected **Caller** is **VIP** in Incident table, then **Impact** and **Urgency** field values setup
1-High.Impact-High and **Urgency High** then **Priority 1- Critical** set automatically
2. **Impact** and **Urgency** fields should be read only on above condition
3. Display on alert message on browser that is (**Caller is VIP!**)

Developer TIP: Make one **Caller VIP (Fred Luddy)** from user record (Already we know how to do this)

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

The screenshot shows the 'Client Scripts' configuration screen. A new script is being created with the following details:

- Name:** VIP Caller Validator (highlighted with a red box)
- Table:** Incident [incident] (highlighted with a red box)
- UI Type:** All
- Type:** onChange (highlighted with a red box)
- Field name:** Caller
- Application:** Global
- Active:**
- Inherited:**
- Global:**

Description:

Requirement

1. If Selected Caller is VIP in Incident table, then Impact and Urgency field values setup
1-High.Impact-High and **Urgency High** then **Priority 1- Critical** set automatically
2. Impact and Urgency fields should be read only on above condition
3. Display on alert message on browser that is (**Caller is VIP!**)

Developer TIP: Make one Caller VIP (Fred Luddy) from user record (Already we know how to do this)

4. Script

Script

```

1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2    if (isLoading || newValue === '') {
3      return;
4    }
5
6    var caller = g_form.getReference('caller_id');
7
8    if(caller.vip == 'true'){
9      alert('Caller is VIP');
10     g_form.setValue('impact', 1);
11     g_form.setValue('urgency', 1);
12     g_form.setDisabled('impact',true);
13     g_form.setDisabled('urgency',true);
14
15  }else{
16
17  if(caller.vip == 'false'){
18
19    g_form.setValue('impact', 3);
20    g_form.setValue('urgency', 3);
21    g_form.setDisabled('impact',false);
22    g_form.setDisabled('urgency',false);
23
24  }
25
26 }

```

5. Open New Incident form check the validation
6. Incident → Click on Create New
7. Select any **VIP Caller** like **Fred Luddy**
8. Check validation on below screen shot for reference

Number	INC0010090	Contact type	-- None --
* Caller	Fred Luddy	State	New
Category	Inquiry / Help	Impact	1 - High 2 - Medium 3 - Low
Subcategory	- None --	Urgency	1 - High 2 - Medium 3 - Low
Service		Priority	1 - Critical 2 - Medium 3 - Low

Exercise: 6

Requirement

1. When customer selected Incident **Category** is **Inquiry/Help**, then **Subcategory** field should be hide
2. Remove choice **1-High** from **Impact** and **Urgency** fields
3. Display an **Error Message** below impact field (**Low impact is not preferring high priority**)

Client Script Type

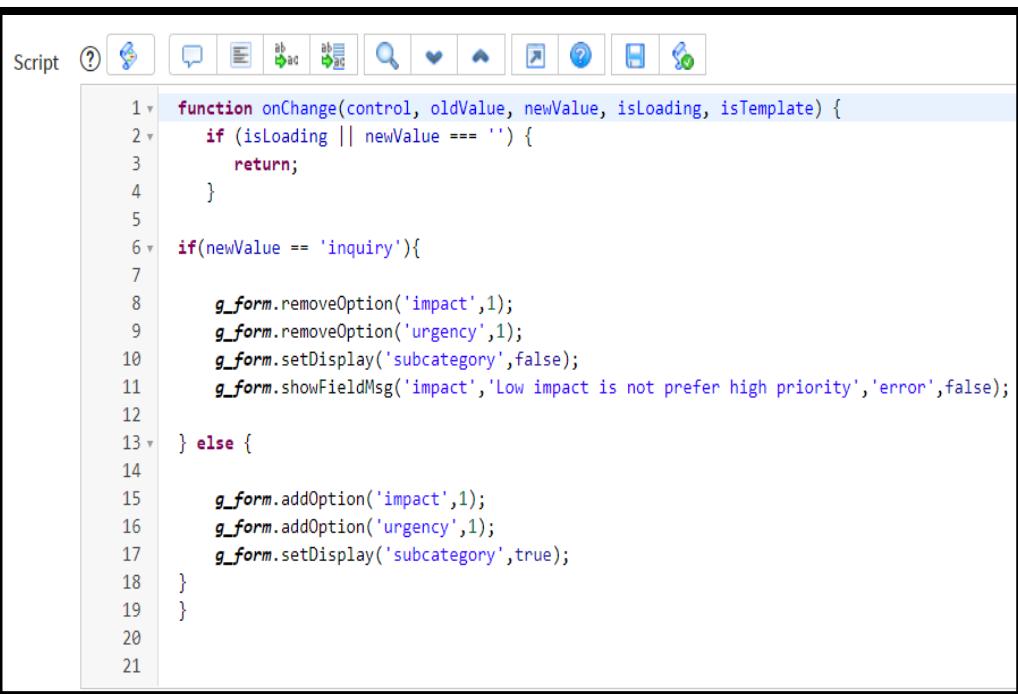
OnChange

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

Name	Category is Enquiry Hide Sub	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	Category		
Description	Requirement 1. If customer selected Incident Category is Inquiry/Help, then Subcategory field should be hide 2. Remove choice 1-High from Impact and Urgency fields 3. Display an Error Message below impact field (Low impact is not prefer high priority)		

4. Script



The screenshot shows a script editor window with a toolbar at the top containing various icons for file operations, search, and help. The main area contains a script function:

```
1 v   function onChange(control, oldValue, newValue, isLoading, isTemplate) {  
2 v     if (isLoading || newValue === '') {  
3       return;  
4     }  
5  
6 v     if(newValue == 'inquiry') {  
7  
8       g_form.removeOption('impact',1);  
9       g_form.removeOption('urgency',1);  
10      g_form.setDisplay('subcategory',false);  
11      g_form.showFieldMsg('impact','Low impact is not prefer high priority','error',false);  
12  
13 v   } else {  
14  
15     g_form.addOption('impact',1);  
16     g_form.addOption('urgency',1);  
17     g_form.setDisplay('subcategory',true);  
18   }  
19 }  
20  
21 }
```

Exercise: 7

Requirement

1. When current incident state is **New** then remove all choices from **State** fieldChoices need to be removed (**In-Progress**, **On-Hold**, **Resolved**, **Closed**, **Cancelled**).
2. When current incident **State** is **In-Progress** or **On-Hold** then remove all choices from **State** Field Choices need to be removed (**New**, **Closed**).
3. When current incident State is **Resolved**, then remove these all choices Choices need to be removed (**New**, **In-Progress**, **On-Hold**, **Cancelled**)
4. When current incident **State** is **Closed**, then state field should be **Read-only**

Client Script Type

OnLoad

Procedure

1. Navigate to System Definition→Client Scripts
2. Open Client Scripts
3. Click on New

Name	State Choice Validation	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onLoad	Global	<input checked="" type="checkbox"/>
Description	<p>Requirement</p> <ol style="list-style-type: none">1. When current incident state is New then remove all choices from State field Choices need to be removed (In-Progress, On-Hold, Resolved, Closed, Cancelled).2. When current incident State is In-Progress or On-Hold then remove all choices from State field Choices need to be removed (New, Closed,).3. When current incident State is Resolved, then remove these all choices Choices need to be removed (New, In-Progress, On-Hold, Cancelled)4. When current incident State is Closed, then state field should be Read-only		

4.Script

```
1+ function onLoad() {
2
3     var inc_state = g_form.getValue('state');
4
5     if(inc_state == 1){
6
7         g_form.removeOption('state',2);
8         g_form.removeOption('state',3);
9         g_form.removeOption('state',6);
10        g_form.removeOption('state',7);
11        g_form.removeOption('state',8);
12
13    }else
14
15    if(inc_state == 2 || inc_state == 3){
16
17        g_form.removeOption('state',1);
18        g_form.removeOption('state',7);
19
20    } else
21
22    if(inc_state == 6){
23
24        g_form.removeOption('state',1);
25        g_form.removeOption('state',2);
26        g_form.removeOption('state',3);
27        g_form.removeOption('state',8);
28
29    } else{
30
31        if(inc_state==7){
32
33            g_form.setDisabled('state',true);
34
35        }
36    }
}
```

1. Open New Incident form check the validation
2. Incident → Click on Create New
3. If current incident state is **New**, then rest of states should not be seen
4. If current incident state is **In-Progress** or **On-Hold**, then **New** and **Closed** States should not visible
5. In **Resolved** state display only **Closed** state value
6. If state is **Closed** State filed should be **Read-only**

Exercise: 8

Requirement:

When incident state is **Closed** then all fields make **read-only**

1. If incident state is **closed**, assigned to, assignment group, category, subcategory, impact, urgency, contact type, short description, caller field are make read-only
2. **Attachment Icon** should not see in **Closed** record
3. All **Related Lists** should be hide

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

A screenshot of the 'New Client Script' configuration window. The window has a black border and contains the following fields:

Name	State Closed Make All field Re	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	State		
Description	<p>Requirement: If incident state is Closed then all fields make read-only</p> <ol style="list-style-type: none">1. If incident state is closed, assigned to, assignment group, category, subcategory, impact, urgency, contact type, short description, caller field are make readonly2. Attachment Icon should not seen in Closed record3. All Related Lists should be hide		

4. Script

Script

```

1 v  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2 v    if (isLoading || newValue === '') {
3      return;
4    }
5
6 v    if(newValue == 7){
7
8      g_form.setDisabled('caller_id',true);
9      g_form.setDisabled('short_description',true);
10     g_form.setDisabled('category',true);
11     g_form.setDisabled('subcategory',true);
12     g_form.setDisabled('impact',true);
13     g_form.setDisabled('urgency',true);
14     g_form.setDisabled('assigned_to',true);
15     g_form.setDisabled('assignment_group',true);
16     g_form.setDisabled('contact_type',true);
17     g_form.hideRelatedLists(); // Hide Related Lists
18     g_form.disableAttachments(); // Hide Attachment Icon
19   }
20
21 }

```

1. Open **New Incident** form check the validation
2. **Incident** → Click on **Resolved** Module
3. Change the state value to **Closed** then some field are display in **Read-only**
4. We couldn't see **Attachment Icon** and **Related Lists**

Manage Attachments (1): Pasted Image [rename] [download]

Number	INC0000055	Contact type	Phone
Caller	Carol Coughlin	State	Closed
Category	-- None --	Impact	1 - High
Subcategory	-- None --	Urgency	1 - High
Service	<input type="text"/>	Priority	1 - Critical
Configuration item	SAP Sales and Distribu	Assigned to	Beth Anglin
Assignment group	Service Desk		

Exercise: 9

Requirement:

When customer changed incident **Priority** field value then work notes should be mandatory

1. If record is **New** or Form **isLoading** or **oldValue == newValue** same then work notes field should not mandatory
2. When customer changed **Priority 1** to **2** then work notes field make **Mandatory**

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition** → **Client Scripts**
2. Open **Client Scripts**
3. Click on **New**



Name	Worknotes Mandatory When	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	Priority		
Description	Requirement: When customer changed incident Priority field value then work notes should be mandatory 1. If opened record is New or oldValue == newValue same then work notes field should not mandatory 2. When customer changed Priority 1 to 2 then work notes field make Mandatory		

4. Script

Script

```

1  function onChange(control, oldValue, newValue, isLoading,
2      isTemplate) {
3          if (isLoading || newValue === '' || g_form.isNewRecord())
4              return;
5
6          else if(newValue == oldValue)
7
8              g_form.setMandatory('work_notes',false);
9          else
10             g_form.setMandatory('work_notes',true);
11             g_form.addInfoMessage('If we change priority value
12 worknotes field mandatory');
13         }

```

5. Open **New Incident** form check the validation
6. **Incident** → Click on **Create New Module**
7. Change the **Priority** to 1 then **Save it**
8. Again change the **Priority** to 2 then get info message to fill **work notes** mandatory

① If we change priority value worknotes field mandatory

② The following mandatory fields are not filled in: Work notes

Category	Inquiry / Help	Impact	1 - High
Subcategory	-- None --	Urgency	1 - High
Service		Priority	1 - Critical

Notes* Related Records Resolution Information

Watch list Work notes list

* Work notes Work notes

Exercise: 10

Requirement

User restriction, not allowing update the cell value from Incident List View

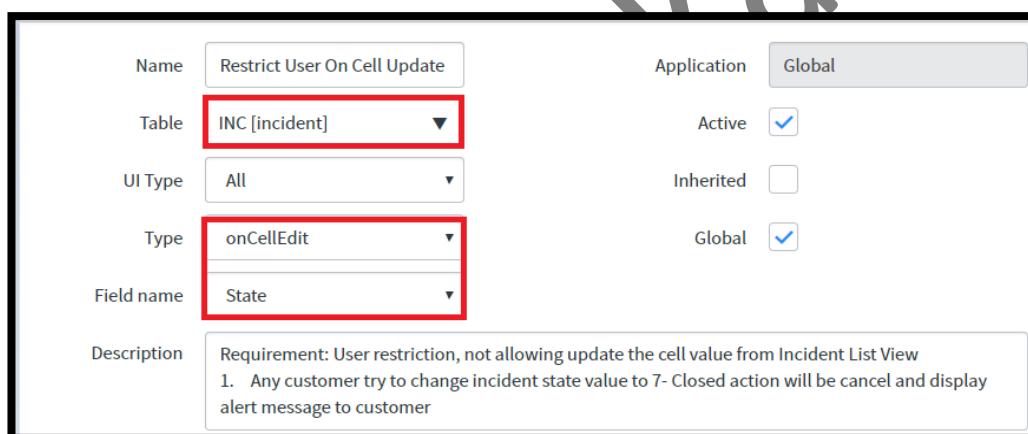
1. Any customer tries to change incident state value to 7- **Closed** action will be cancel and display alert message to customer
2. Alert Message (**System not allow user to update cell value**)

Client Script Type

OnCellEdit

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Open **Client Scripts**
3. Click on **New**



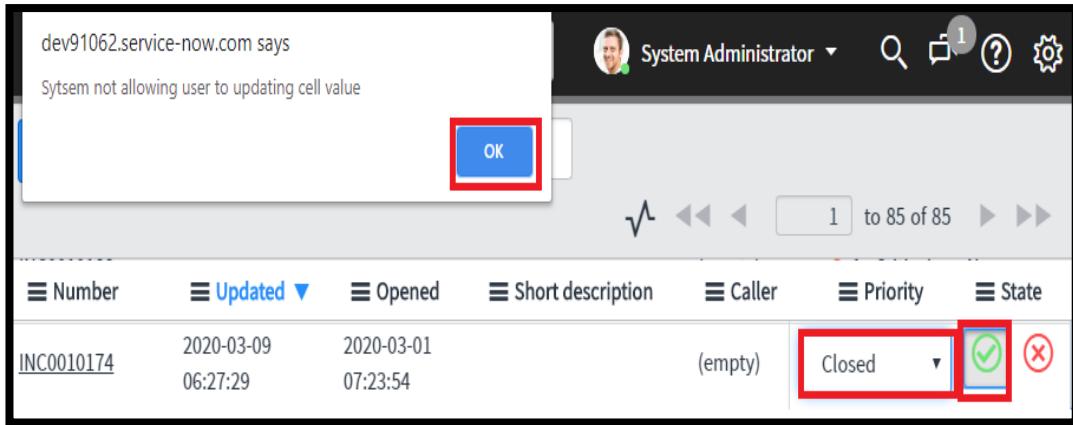
4.Script

```
function onCellEdit(sysIDs, table, oldvalues, newValue, callback) {
    var saveAndClose = true;
    //System cant allow user to update cell value

    if(newValue == 7){
        alert('Sytsem not allowing user to update cell value');
        saveAndClose = false;
    } else {
        saveAndClose = true;
    }

    callback(saveAndClose);
}
```

4. Open **Incident List View** and check the validation
5. **Incident** → Click on **All** Module
6. Change the state value to **7-Closed**
7. Get Error Message (System not allowing user to update cell value)



Exercise: 11

Requirement:

1. When user submit incident form need to display **Assigned to** Name as alert message
2. Give confirmation message also

Client Script Type

OnSubmit

Procedure

1. Navigate to **System Definition**→**Client Scripts**
2. Open Client Scripts
3. Click on **New**

The screenshot shows the 'Client Scripts' dialog with the following configuration:

Name	Assigned to alert on Form Su	Application	Global
Table	INC [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onSubmit	Global	<input checked="" type="checkbox"/>

Description:

Requirement:

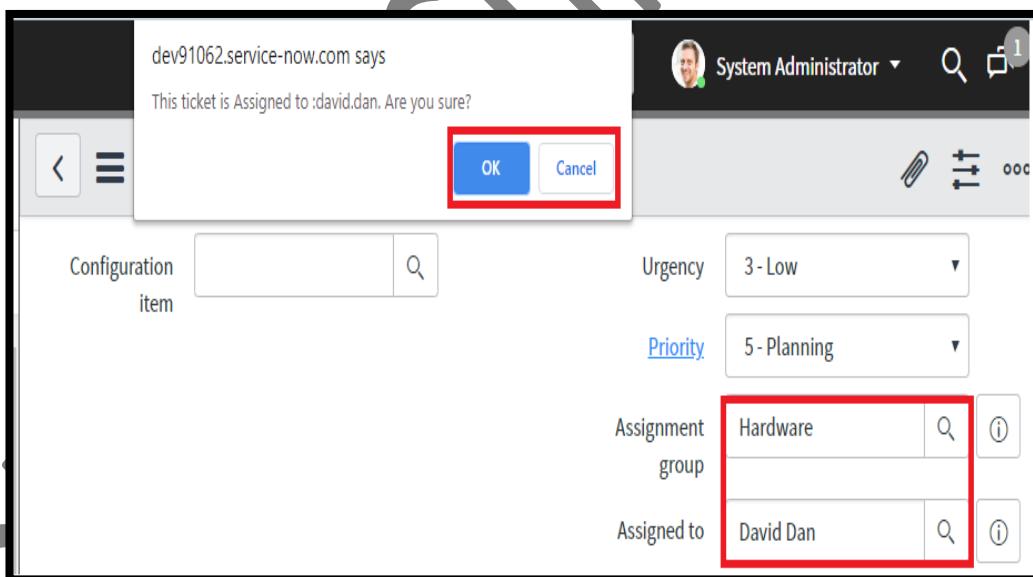
1. When user submit incident form need to display Assigned to Name as alter message
2. Give confirmation message also

4. Script

Script

```
function onSubmit() {  
    var user = g_form.getReference('assigned_to');  
  
    var ticket = confirm('This ticket is Assigned to :'+ user.user_name +'. Are you sure?');  
  
    if(ticket == true){  
        return false ;  
    }  
}
```

5. Open **New Incident** check the validation
6. Fill **Assigned to** field and **Submit** form
7. Display alert message and given confirmation message



Exercise: 12

Requirement:

Auto assign the value to **Assignment Group** depends on category

1. If **Category** is **Network** auto assigned **Network Assignment Group**
2. Category is **Hardware** auto assigned to **Hardware Assignment Group**
3. Category is **Database** auto assigned to **Database Assignment Group**
4. Category is **Software** auto assigned to **Software Assignment Group**

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition**→**Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

The screenshot shows the 'Client Scripts' configuration screen. A new script is being created with the following details:

Name	Auto assignment to group up	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	Category		
Description	Requirement: Auto assign the value to Assignment Group depends on category 1. If Category is Network auto assigned Network Assignment Group 2. Category is Hardware auto assigned to Hardware Assignment Group 3. Category is Database auto assigned to Database Assignment Group 4. Category is Software auto assigned to Software Assignment Group		

4. Script

Script

```

1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2    if (isLoading || newValue === '') {
3      return;
4    }
5
6    if (newValue == 'software') {
7      g_form.setValue('assignment_group', 'software');
8
9    } else if (newValue == 'hardware') {
10      g_form.setValue('assignment_group', 'hardware');
11    } else if (newValue == 'network') {
12      g_form.setValue('assignment_group', 'network');
13    } else if (newValue == 'database') {
14      g_form.setValue('assignment_group', 'database');
15    }
16  }
17

```

5. Open **New Incident** check the validation
6. Select the **Category** is **Software** then ticket is auto assign to **Software Assignment Group**
7. Select **Remaining Category** Also Similarly check the validation

The form is a 'New Incident' submission page. It contains the following fields:

- Number:** INC0010070
- Contact type:** -- None --
- * Effected User:** Abel Tuter
- * Category:** Software (highlighted with a red box)
- Subcategory:** -- None --
- Impact:** 3 - Low
- Service:** (empty field)
- Urgency:** 3 - Low
- Priority:** 5 - Planning
- Configuration item:** (empty field)
- Assignment group:** Software (highlighted with a red box)

Exercise: 13 Table Incident

Requirement: Auto set **Caller** value into caller field depends on current logged in user

1. Display alert message say **Hello** message with logged in user full name
2. Caller field should be Read-Only

Client Script Type

OnLoad

Procedure

1. Navigate to **System Definition** → **Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

Name: Caller Should be set Automat

Table: Incident [incident]

UI Type: All

Type: onLoad

Application: Global

Active:

Inherited:

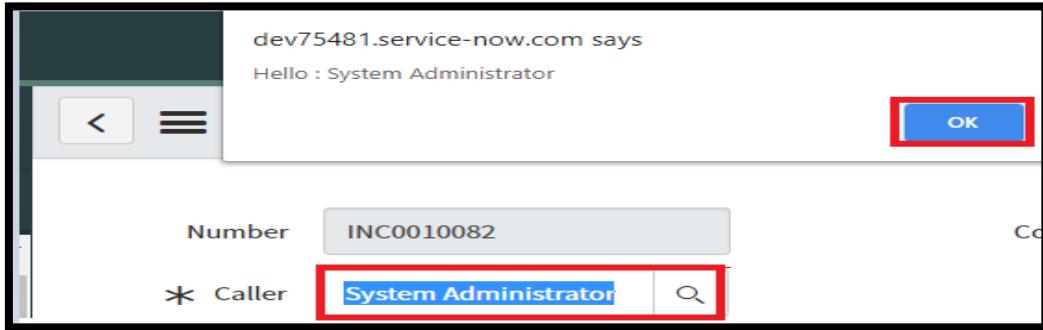
Global:

Description: Auto set caller field value depends on current loged in user

4. Script

```
function onLoad() {
    var currentUser = g_user.userID;
    g_form.setValue('caller_id', currentUser);
    alert('Hello : ' + g_user.fullName);
}
```

5. Open **New Incident** check the validation
6. Click on **Create New** module
7. Click on **Ok** button



Exercise: 14

Requirement

Date Validation between two fields for Incident table

1. Create **Two custom fields** in incident tables (**Start Date, End Date**)
2. **Start date** should be always before End date
3. **Start date** should not equal to **End date**
4. Give me **Error Message** under End date field

Client Script Type

OnChange

Procedure

1. Navigate to **System Definition → Client Scripts**
2. Open **Client Scripts**
3. Click on **New**

Name	Start and End Date Validator	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	End Date		
Description	Requirement: Date Validation between two fields in Incident table 1. Create Two custom fields in incident tables (Start Date, End Date) 2. Start date should be always before End date 3. Start date should not equal to End date 4. Give me Error Message under End date field		

4. Script

Script           

```

1 function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2     if (isLoading || newValue === '') {
3         return;
4     }
5
6     var startDate = g_form.getValue('u_start_date');
7     var endDate = g_form.getValue('u_end_date');
8
9     if (startDate > endDate || startDate == endDate) {
10        g_form.showFieldMsg('u_end_date', 'Start date should not be
greater than and equal to end date', 'error', false);
11    }
12
13}
14

```

Activate Window
Go to Settings to act

5. Open **New Incident** check the validation
6. Click on **Create New** module
7. Select Start and End date (**Start date** should be after **End date**) check error message
8. Check another validation both dates should not equal

Subcategory	-- None --	Urgency	3 - Low
Service	<input type="text"/>	Priority	5 - Planning
Configuration item	<input type="text"/>	Assignment group	<input type="text"/>
Start Date	2020-03-12	* Assigned to	System Administrator
End Date	2020-03-02	Phone Number	<input type="text"/>
Start date should not be greater than end date			

Exercise: 15

Requirement:

Hide Resolution Information section when state is not equal to Resolved

- When Page Is Loading and state is not equal to Resolved then hide Resolution Information section otherwise display section

Client Script Type

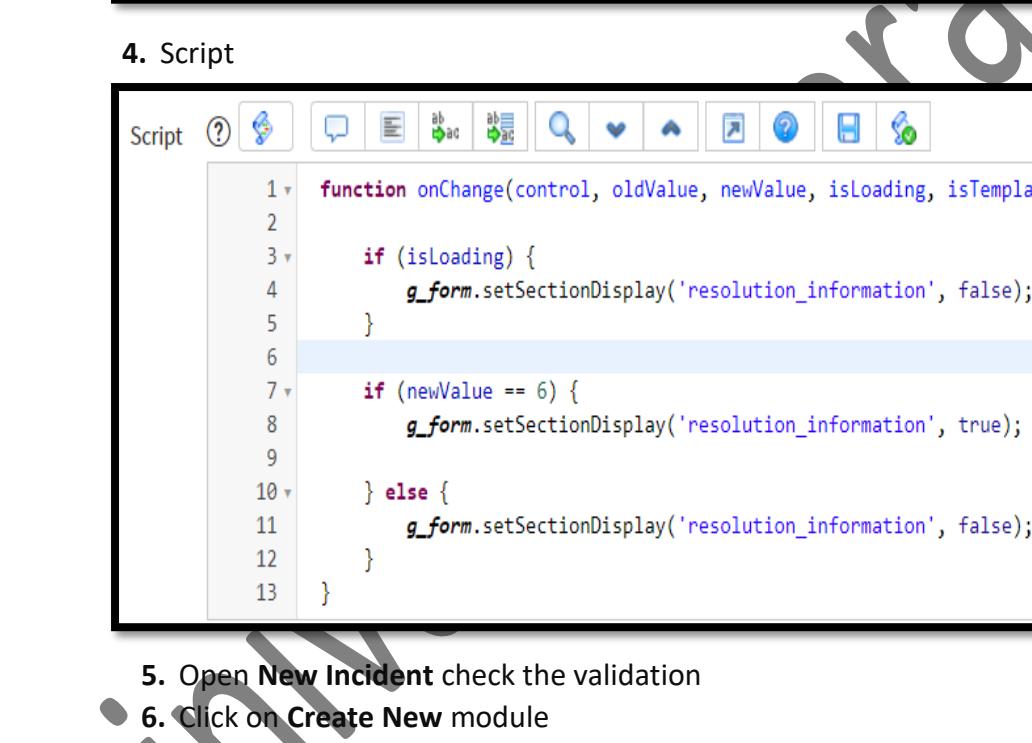
OnChange

Procedure

- Navigate to System Definition→Client Scripts
- Open Client Scripts
- Click on New

Name	Secton Hiding Based on State	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
UI Type	All	Inherited	<input type="checkbox"/>
Type	onChange	Global	<input checked="" type="checkbox"/>
Field name	State		
Description	Requirement: Hide Resolution Information section when state is not equal to Resolved 1. When Page Is Loading and state is not equal to Resolved then hide Resolution Information section otherwise display section		

4. Script



```

Script ? 
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading) {
        g_form.setSectionDisplay('resolution_information', false);
    }
    if (newValue == 6) {
        g_form.setSectionDisplay('resolution_information', true);
    } else {
        g_form.setSectionDisplay('resolution_information', false);
    }
}

```

5. Open New Incident check the validation

6. Click on **Create New** module

7. When form is Loaded and state not equal to **Resolved** hide the Resolution Information Section

Number	INC0010158	Contact type	-- None --
* Caller	<input type="text"/>	State	In Progress
Category	None	Impact	3 - Low
Notes	Related Records	Section Hide	
Watch list		Work notes list	

Lesson-10

UI Actions

Agenda

- UI Actions Overview
- UI Actions Controls
- UI Actions on Server Side
- UI Actions on Client Side
- UI Actions Exercises
- Sending Notifications Through UI Actions

Srinivas

UI Action Overview

UI Actions defines Buttons, Hyperlinks, list menus and form context menus

Configure UI actions to make the UI more interactive, customized, and specific to user activities. **Service now Administrator** and **UI Action Admin** will create UI actions in service now platform

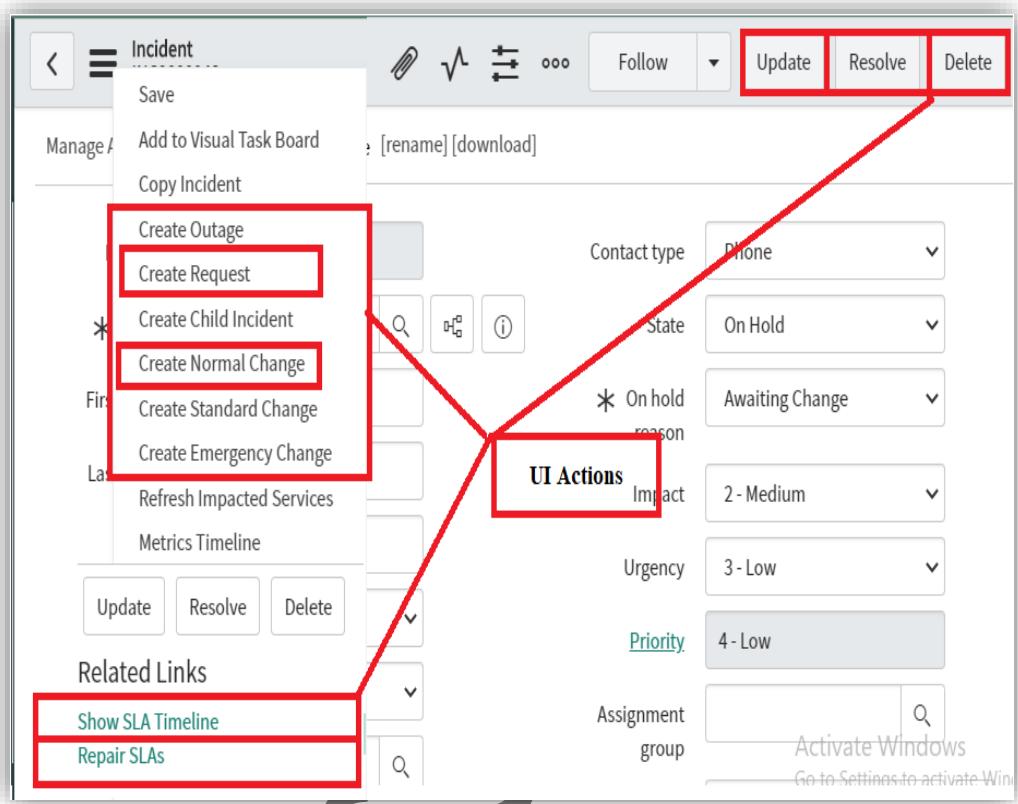
When developer develops some scripts to automate business logic, we try to automate how the logic triggers. If logic should run when a record is created or is set to a certain state, I use a Business Rule to automatically execute a script. Sometimes a user needs to initiate the work. UI Actions provide the interface elements for a user to take action from a form or list. Configure UI Actions as buttons or links to create interface elements with which users can interact.

UI Action Controls

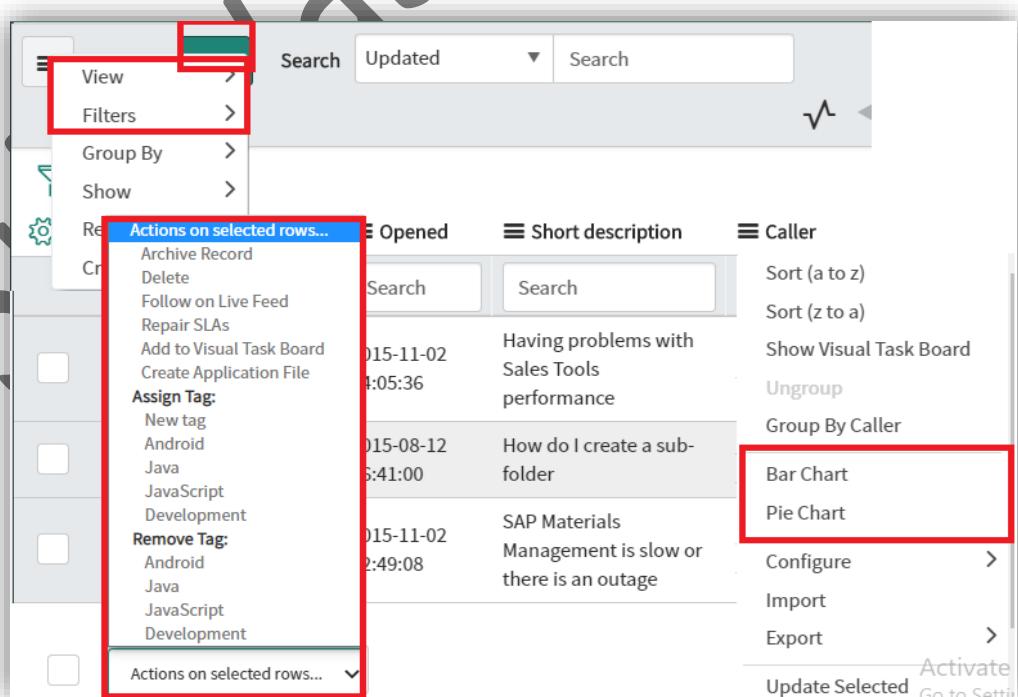
We can create UI Actions for these all places:

- A button on a form View.
- A context menu item on a form that appears when you open the form context menu or right-click the form header.
- A related link in a form.
- A button in the banner on top of a list.
- A button at the bottom of a list.
- A context menu item on a list that appears when you open the list context menu or right-click the list header.
- A menu item for the action choice list at the bottom of a list.
- A related link at the bottom of a list.

Form UI Actions



List UI Action



UI Action on Server Side

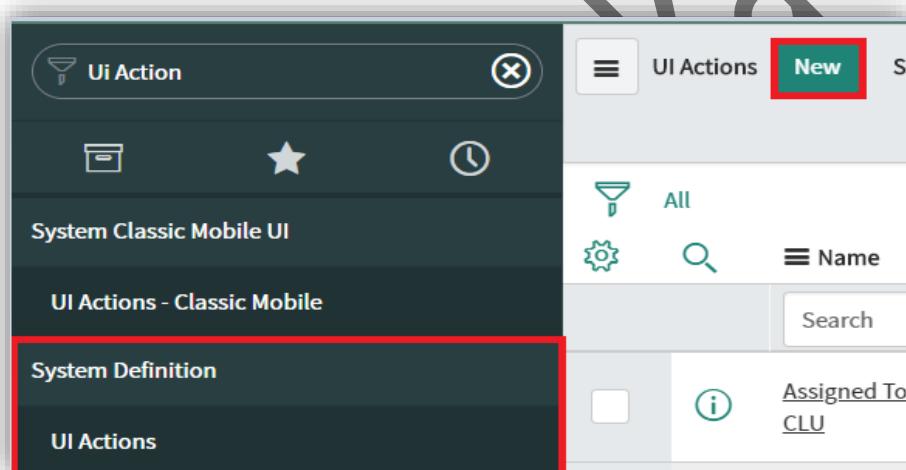
Service now **Admin** or **Developer** can create new UI Action or modify existing one

Exercise-1

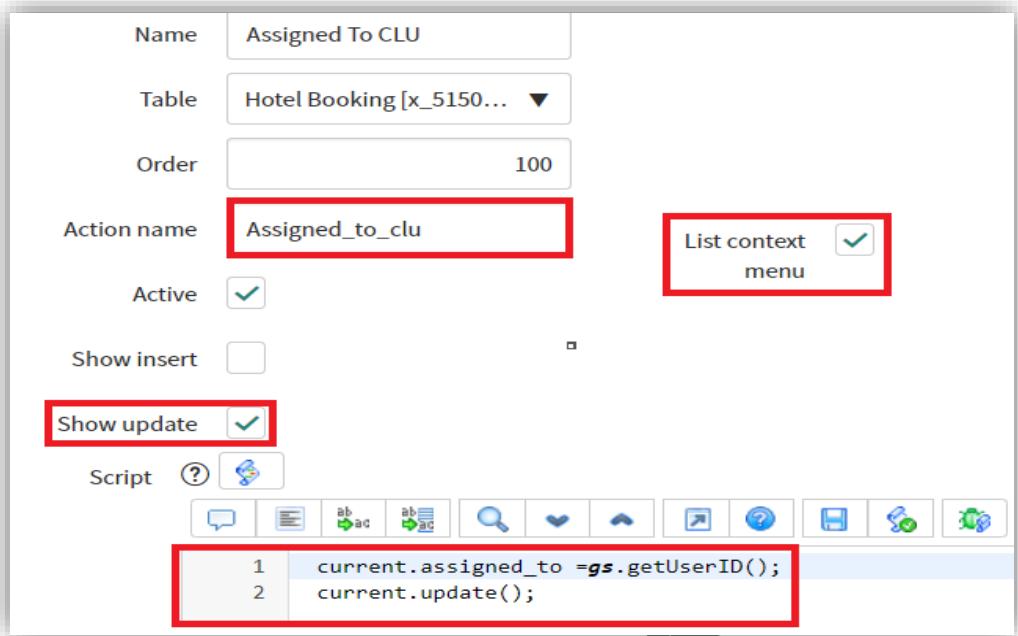
There should be a button on Incident form to assigned ticket to current logged in user

Procedure

1. Navigate to **System Definition** → **UI Actions**
2. Click on **New**



3. Fill UI Action form
4. Name: Assigned to CLU
5. Table: Incident
6. Order:100
7. Action Name: Assigned_to_clu
8. Show Update: Checked
9. List Context Menu :Checked
10. Active: Checked



9. Script

```
current.assigned_to = gs.getUserID();
current.update();
```

10. Click on **Submit**

11. Check Your Code

The screenshot shows the 'Hotel Bookings' list view. A context menu is open over a row, with the 'Assigned To CLU' option highlighted.

- Show Matching
- Filter Out
- Show After
- Show Before
- Copy URL to Clipboard
- Copy sys_id
- Assign Tag >
- Assigned To CLU**

UI Action on Client Side

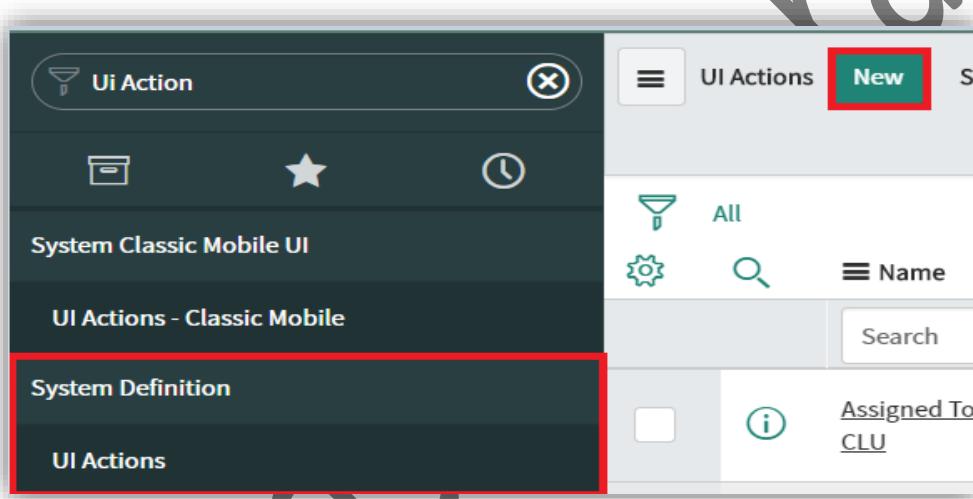
UI action can run on client side also following below example

Exercise-2

There should be a button on Incident form to assign the ticket to current logged in user

Procedure

1. Navigate to System Definition → UI Actions
2. Click on New



3. Fill UI Action form
4. **Name:** Auto Assignment Incident
5. **Table:** Incident
6. **Order:** 790
7. **Action Name:** Auto_assignment
8. **Show Insert:** Checked
9. **Show Update:** Checked
10. **Client:** True
11. **List v2 Compatible:** True
12. **List v3 Compatible:** True
13. **Active:** Checked
14. **Form Button:** True

UI Action
Auto Assignment Incident

Name: Auto Assignment Incident

Table: Incident [incident]

Order: 790

Action name: auto_assignment

Active:

Show insert:

Show update:

Client:

List v2 Compatible:

Application: Global

Form button:

Form context menu:

Form link:

Form style: -- None --

List banner button:

List bottom button:

List context menu:

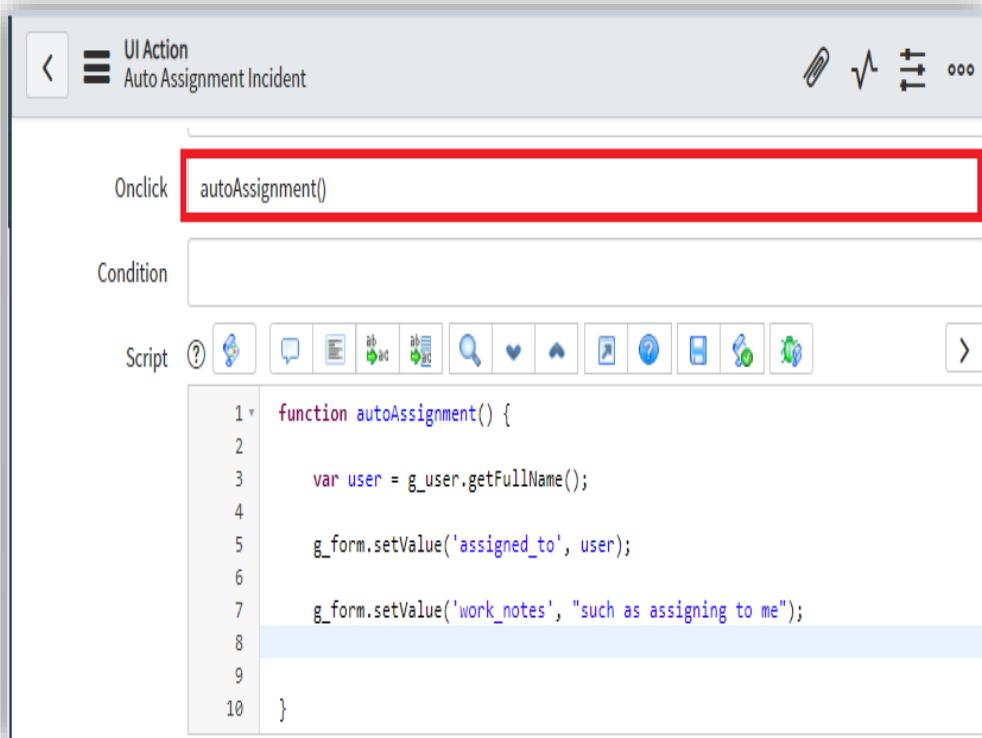
List choice:

15. Onclick: autoAssignment()

16. Script

```
function autoAssignment (){  
    var user = g_user.userID;  
    g_form.setValue('assigned_to', user);  
    g_form.setValue('work_notes', "such as assigning to me");  
    g_form.save();  
}
```

17. Click on **Submit**

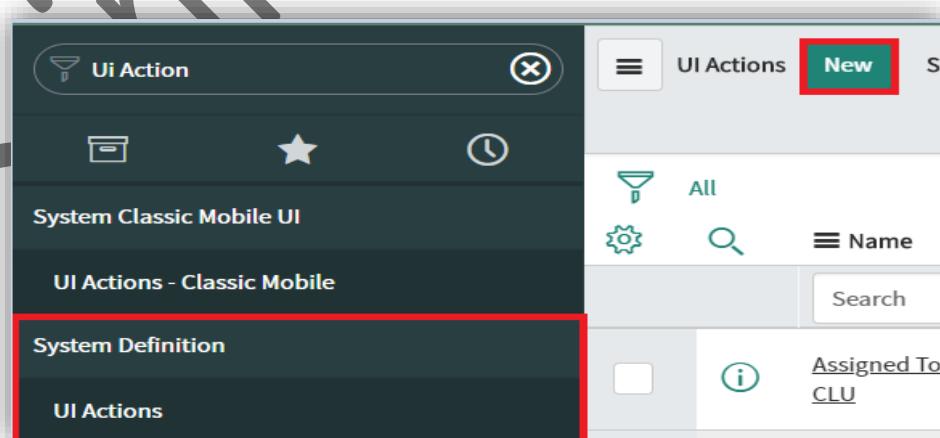


Exercise-3

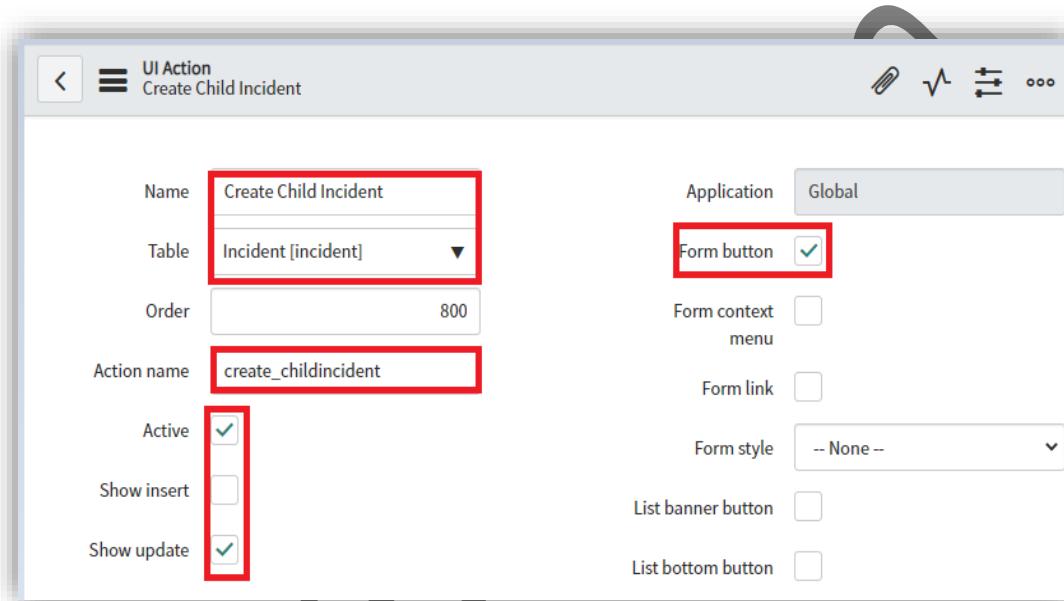
There should be a button on incident form to create new **Child Incident** from being on the Parent incident form. Once created child incident immediately it should be redirect to child incident

Procedure

1. Navigate to System Definition → UI Actions
2. Click on New



3. Fill UI Action form
4. **Name:** Create Child Incident
5. **Table:** Incident
6. **Order:** 800
7. **Action Name:** create_childIncident
8. **Show Update:** Checked
9. **Active:** Checked
- 10. Form button:** True



11. Script

```

var inc = new GlideRecord("incident");
inc.initialize();
inc.short_description = current.short_description;
inc.category = current.category;
inc.subcategory = current.subcategory;
inc.caller_id = current.caller_id;
inc.opened_by = gs.getUserID();
inc.impact = current.impact;
inc.urgency = current.urgency;
inc.parent = current.sys_id;
var sysID = inc.insert();
gs.addInfoMessage('New Child Incident ' + inc.number + " created");
action.setRedirectURL(inc);
action.setReturnURL(current);

```

Script

```

1 var inc = new GlideRecord("incident");
2 inc.initialize();
3 inc.short_description = current.short_description;
4 inc.category = current.category;
5 inc.subcategory = current.subcategory;
6 inc.caller_id = current.caller_id;
7 inc.opened_by = gs.getUserID();
8 inc.impact = current.impact;
9 inc.urgency = current.urgency;
10 inc.parent = current.sys_id;
11 var sysID = inc.insert();
12 gs.addInfoMessage("New Child Incident " + inc.number + " created");
13 action.setRedirectURL(inc);
14 action.setReturnURL(current);
15

```

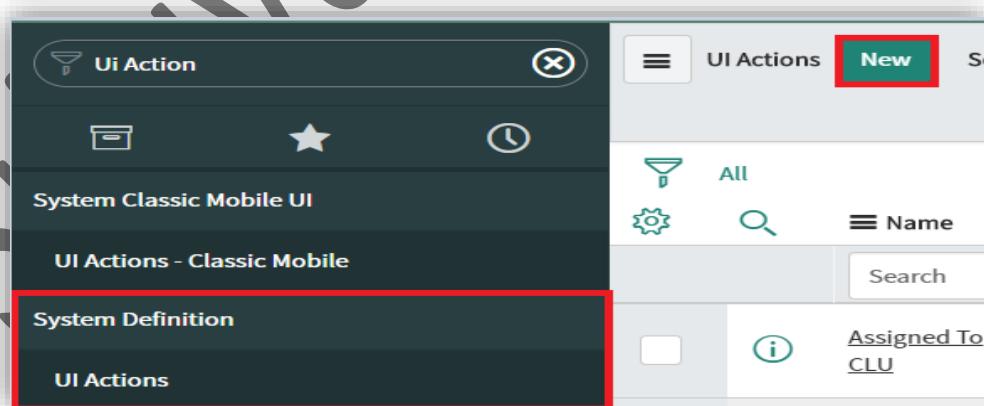
12. Click on Submit

Exercise-4

There should be a UI action on incident record to create 3 incident tasks automatically associated with same incident.

Procedure

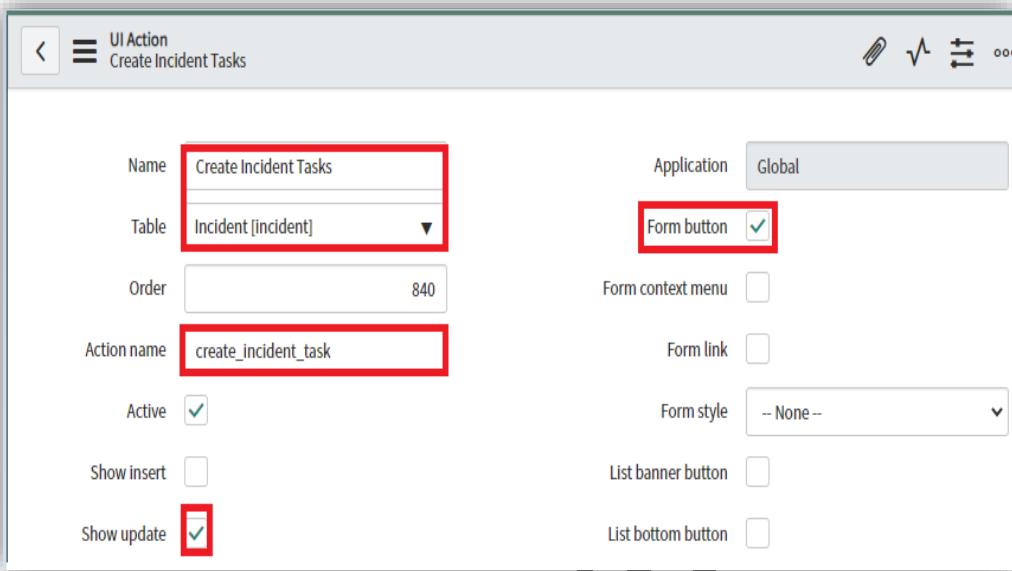
1. Navigate to System Definition → UI Actions
2. Click on New



3. Fill UI Action form
4. **Name:** Create Incident Tasks
5. **Table:** Incident
6. **Order:** 840
7. **Action Name:** create_incident_task
8. **Show Update:** Checked

9. Active: Checked

10. Form button: True



11. Script

```
var shortDesc = ['Incident Task1', 'Incident Task2', 'Incident Task3'];
for (i = 0; i < shortDesc.length; i++) {

    var gr = new GlideRecord('incident_task');
    gr.initialize();
    gr.short_description = shortDesc[i];
    gr.description = current.short_description;
    gr.incident = current.sys_id;
    gr.insert();
}

action.setRedirectURL(current);
```

Script

```

1 var shortDesc = ['Incident Task1', 'Incident Task2', 'Incident Task3'];
2
3 for (i = 0; i < shortDesc.length; i++) {
4
5     var gr = new GlideRecord('incident_task');
6     gr.initialize();
7     gr.short_description = shortDesc[i];
8     gr.description = current.short_description;
9     gr.incident = current.sys_id;
10    gr.insert();
11 }
12 action.setRedirectURL(current);

```

12. Click on **Submit**

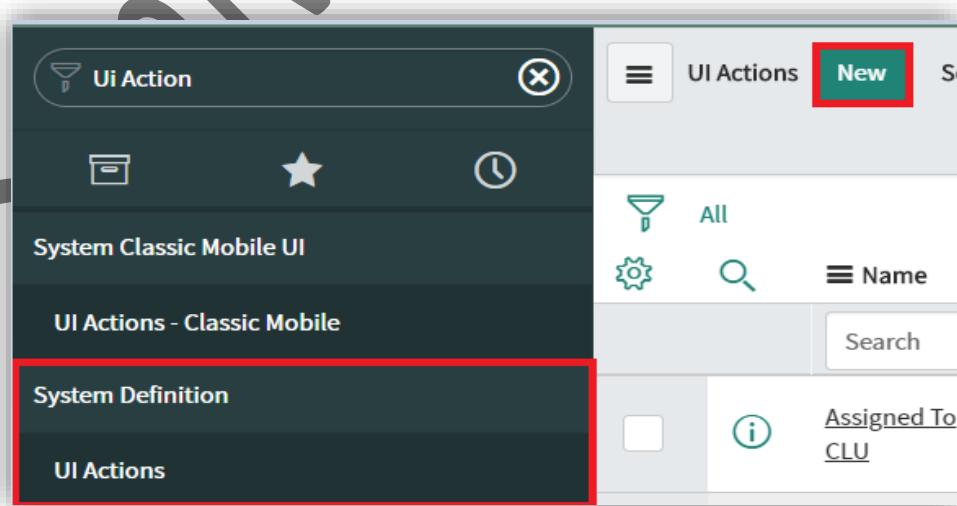
13. Check the result

Exercise-5

Create new UI Action to open problem form from Incident table record

Procedure

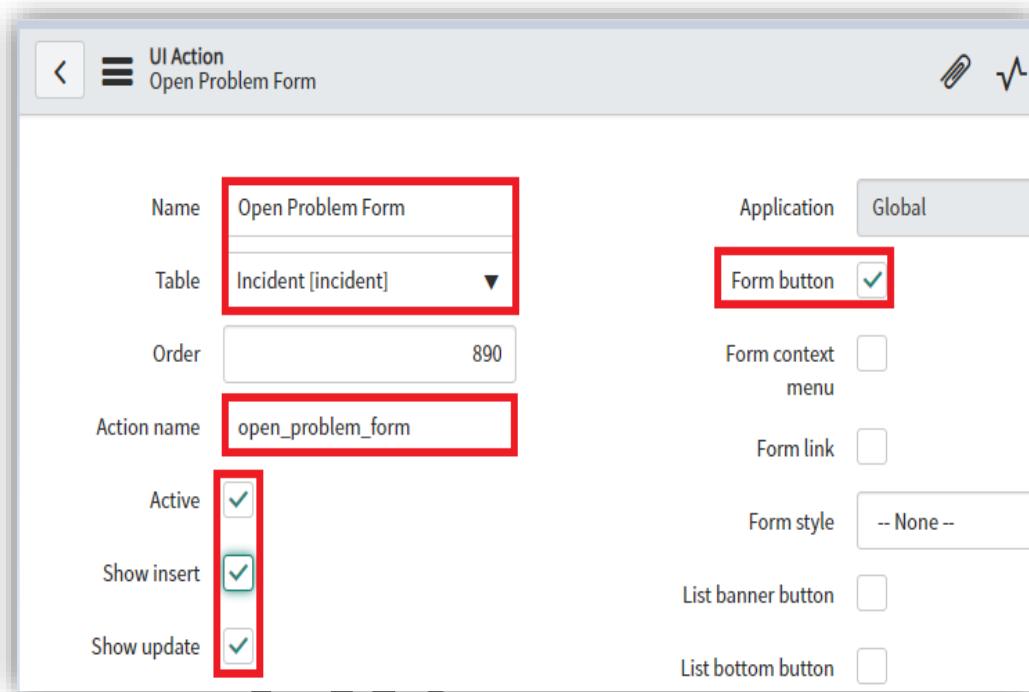
1. Navigate to **System Definition** → **UI Actions**
2. Click on **New**



3. Fill UI Action form

4. **Name:** Open Problem Form

5. **Table:** Incident
6. **Order:** 890
7. **Action Name:** open_problem_form
8. **Show Insert:** Checked
9. **Show Update:** Checked
10. **Active:** Checked
11. **Form button:** True



12. Script

```
var url = '/problem.do?sys_id=-1';
action.setRedirectURL(url);
```

Script

①


```
1 var url = '/problem.do?sys_id=-1';
2 action.setRedirectURL(url);
3 |
```

13. Click on Submit

Exercise-6

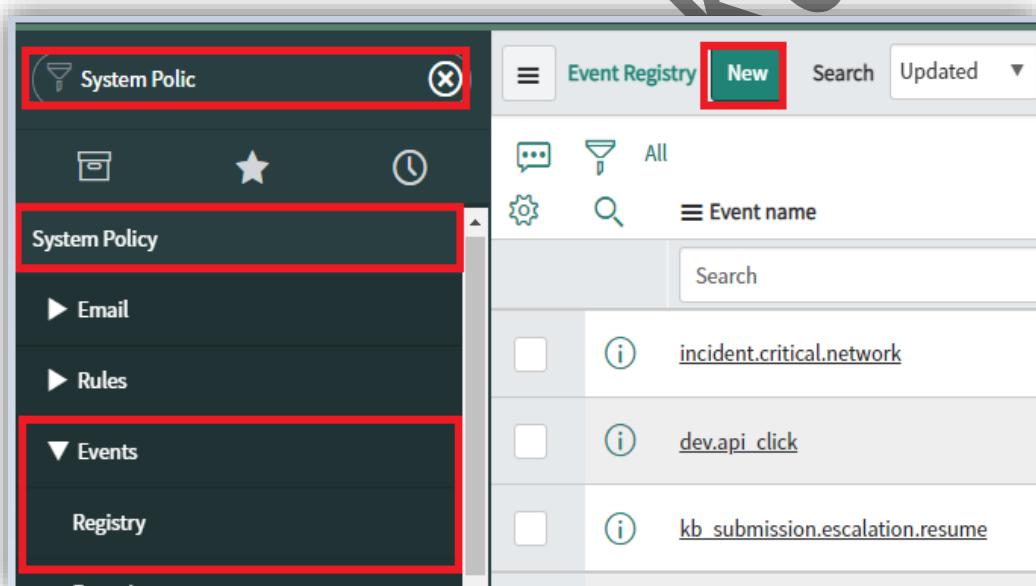
Create new UI Action to send notification to caller about incident state changes

1. Create Event Registry
2. Create Email Notification
3. Create UI Action

Creating Event Registry

Procedure

1. Navigate to System Policy → Events → Registry
2. Click on New



3. **Event name:** Send the notification to assigned to
4. **Table:** Incident
5. **Fired by:** UI Action
6. **Description:** Create new UI Action to send notification to caller about incident state changes
7. Click on **Submit**

Event Registration
Send the notification to assigned to

Event name	send.notification.assignedto	Application	Global
Table	Incident [incident]	Queue	
		Caller Access	-- None --
Fired by	UI Action		
Description	Create new UI Action to send notification to caller about incident state changes		

Creating Notification

Procedure

1. Navigate to **System Notification** → **Email** → **Notification**
2. Click on **New**

The screenshot shows the 'System Notification' interface. On the left, there's a sidebar with categories: 'System Notification' (expanded), 'Email' (selected and highlighted with a red box), 'Digest Intervals', 'Notifications' (highlighted with a red box), and 'Notification Email Scripts'. The main area is titled 'Notifications' and has a 'New' button highlighted with a red box. Below it, there's a search bar and a table listing notifications. The table has columns for 'Name', 'Active', and 'Table'. Two entries are listed:

Name	Active	Table
Email assigned to group (sc_task)	true	Catalog Task [sc_task]
Email assigned to (sc_task)	true	Catalog Task [sc_task]

3. **Name:** State Change to IN progress
4. **Table:** Incident
5. **Active:** True
6. **Category:** Uncategorized

Notification - State change to inprogress [Advanced view*]

Use Notifications to notify users about specific activities in ServiceNow, such as updates to incidents or change requests. administrators to specify

- When to send the notification
- Who receives the notification
- What content is in the notification

Name	State change to inprogress	Type	EMAIL
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
* Category	Uncategorized	Allow Digest	<input type="checkbox"/>

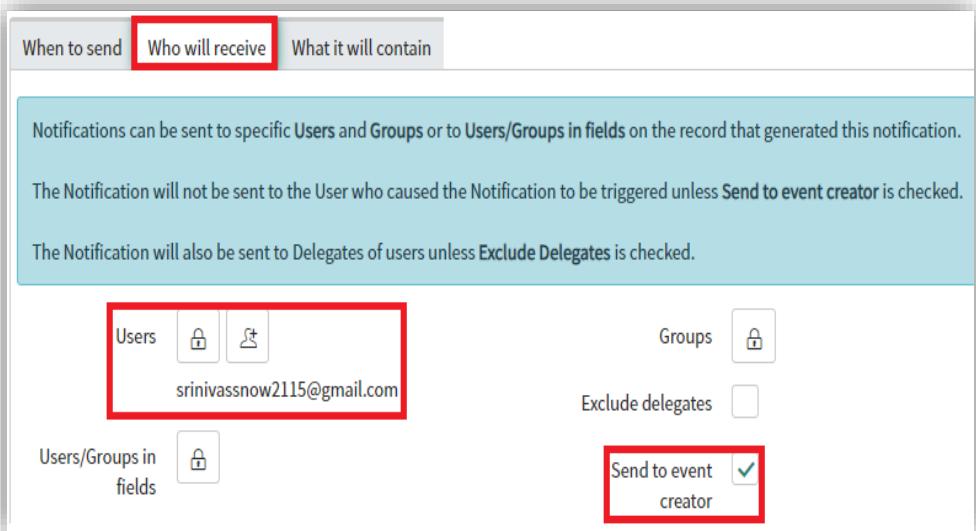
When to send

7. **Send when:** Event is fired
8. **Event name:** send.notification.assignedto

When to send	Who will receive	What it will contain
Send when	Event is fired	Weight
Event name	send.notification.assignedto	
Conditions	Add Filter Condition	Add "OR" Clause
	-- choose field --	-- oper --

9. **Users:** Srinivassnow2115@gmail.com

10. **Send event creator:** true

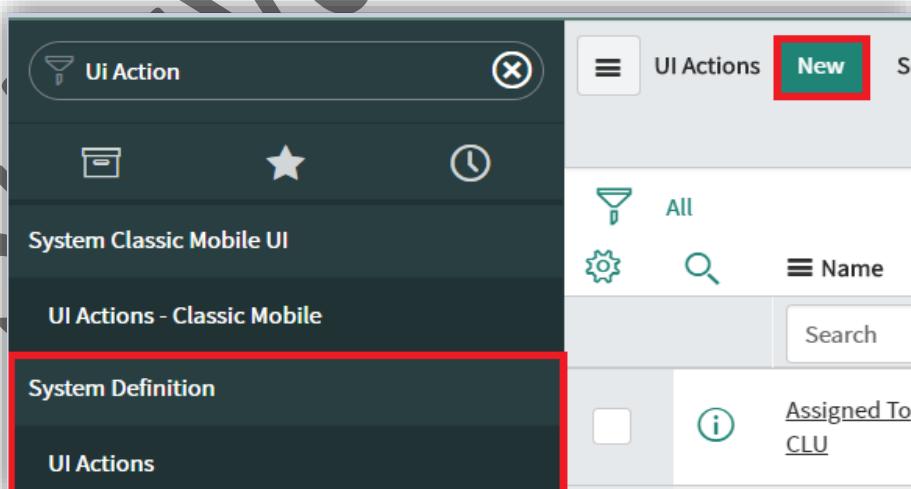


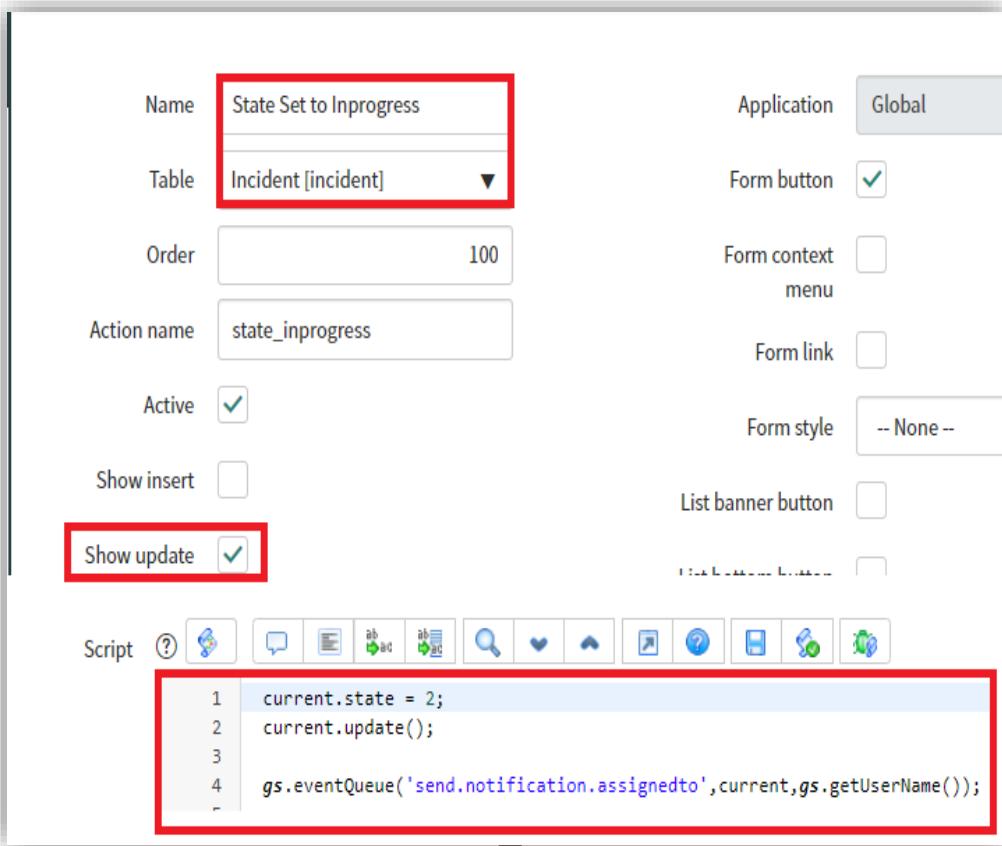
11. Subject: State changes to inprogress state: \${state}
12. Assigned to: \${assigned to}

Creating UI Action

Procedure

1. Navigate to System Definition → UI Action
2. Click on New





3. Fill UI Action form

4. Name: State set to inprogress

5. Table: Incident

6. Order: 100

7. Action Name: state_inprogress

8. Show Update: Checked

9. Active: True

10. Form button: True

11. Script

```
current.state = 2;
current.update();
gs.eventQueue('send.notification.assignedto',current,gs.getUserName());
()
```

12. Click on Submit

UI Actions on Client and Server side

There should be a UI Action on incident table to Resolve incident ticket

Procedure

1. Navigate to **System Definition**→**UI Actions**
2. Click on **New**

The screenshot shows the 'UI Action' configuration dialog for an action named 'Resolve'. The 'Table' is set to 'Incident [incident]'. The 'Order' is 100. The 'Action name' is 'resolve_incident'. The 'Application' is 'Global'. The 'Active' checkbox is checked. The 'Show insert', 'Show update', and 'Client' checkboxes are also checked. The 'Onclick' field contains the JavaScript code 'resovleIncident();'. A large watermark 'Srinivas' is diagonally across the image.

Name	Resolve	Application	Global
Table	Incident [incident]	Form button	<input checked="" type="checkbox"/>
Order	100	Form context menu	<input type="checkbox"/>
Action name	resolve_incident	Form link	<input type="checkbox"/>
Active	<input checked="" type="checkbox"/>	Form style	-- None --
Show insert	<input checked="" type="checkbox"/>	List banner button	<input type="checkbox"/>
Show update	<input checked="" type="checkbox"/>	List bottom button	<input type="checkbox"/>
Client	<input checked="" type="checkbox"/>	List context menu	<input type="checkbox"/>
Onclick	resovleIncident();		

Script

```
function resolveIncident(){
    //Set the 'Incident state' and 'State' values to 'Resolved', and
    display mandatory fields
    g_form.setValue('incident_state', 6);
    g_form.setValue('state', 6);
    g_form.setValue('resolved_by', g_user.userID);

    gsftSubmit(null, g_form.getFormElement(), 'resolve_incident');
    //MUST call the 'Action name' set in this UI Action
}

//Code that runs without 'onclick'
//Ensure call to server-side function with no browser errors

if (typeof window == 'undefined')
    serverResolve();

function serverResolve(){
    current.incident_state = IncidentState.RESOLVED;
    current.state = IncidentState.RESOLVED;
    current.resolved_by = gs.getUserID();
    current.update();
}
```

Condition new global.IncidentUtils().canResolveIncident(current)

Script

```
1 function resolveIncident(){
2     //Set the 'Incident state' and 'State' values to 'Resolved', and display
3     //mandatory fields
4     g_form.setValue('incident_state', 6);
5     g_form.setValue('state', 6);
6     g_form.setValue('resolved_by', g_user.userID);
7
8     gsftSubmit(null, g_form.getFormElement(), 'resolve_incident'); //MUST call
9     //the 'Action name' set in this UI Action
10
11 //Code that runs without 'onclick'
12 //Ensure call to server-side function with no browser errors
13 if (typeof window == 'undefined')
14     serverResolve();
15
16 function serverResolve(){
17     current.incident_state = IncidentState.RESOLVED;
18     current.state = IncidentState.RESOLVED;
19     current.resolved_by = gs.getUserID();
20     current.update();
21 }
```

Srinivas

Lesson-11

Scheduled Jobs

Agenda

- Scheduled Jobs Overview
- Scenarios where we can use schedule jobs
- Scheduled Jobs Tables
- Scheduled jobs states
- Who will create new scheduled job
- Creating New Scheduled Job
- How to share report through schedule job
- Create schedule job template
- Send notification through scheduled Job



Scheduled Jobs Overview

Scheduled Jobs are automated pieces of work that can be performed at a specific time or on a recurring schedule. Schedule jobs perform some operations or actions or some code will execute at specific interval or time. Schedule jobs are running from server side

Scheduled are useful way to automate processes in Service-now and need to remove some of the administrative burden of the tool off of the shoulders of your Service-now administrators. It's very easy to create a scheduled job or a scheduled import.

We can automate the following tasks below

- Automatically generate and schedule an entity of records, such as an incident, change item, configuration item, from a template
- Automatically run and distribute a report
- Generate a record (incident, Problem, change, configuration item, etc.) from a template
- Run a business rule and do whatever the rule contains
- Scheduling at the end of the month
- Scheduling for weekdays
- Executing scheduled jobs from scripts
- Fire an email notification from a schedule job

Some scenarios where we can use schedule jobs in

ServiceNow

- Specific report will be delivered to stakeholder every day, every week, every month or at any specific period.
- Specific task record will get created and assigned to specific team every month in the system automatically.
- The report with email contains complete information will be delivered to stakeholders.
- **Scenario:** Auto closure of tickets such as Incident ticket under state pending customer action, where customer is not responding from last few days.
- **Scenario:** Auto closure of incident ticket when ticket is in resolved state from last 9 months' days.
- We can use schedule job to trigger different events weekly, monthly or at any specific period.

Scheduled Jobs Tables

1. sysauto_script (Scheduled Script Execution)
2. scheduled_import_set (Scheduled Import Sets)
3. sysauto_template (Scheduled Template Generation)
4. sysauto_report (Scheduled Report)

Scheduled job states

A scheduled job can be in any one of the following states.

Ready

Job is ready to run at the next scheduled interval.

Running

Job is in the process of carrying out a task.

Queued

Job has been added to the scheduler queue and is waiting to run.

Error

Error occurred while running the job.

Create a new scheduled job

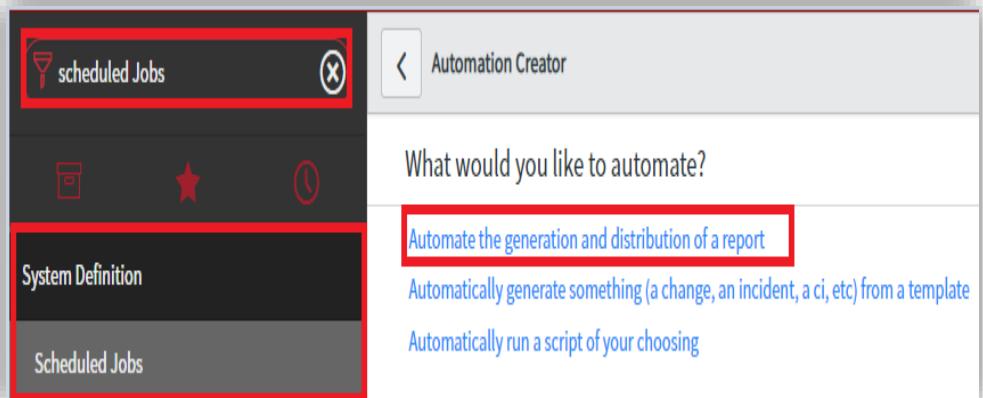
Automate generation and distribution of a report

Exercise: 1

Generate and distribute scheduled reports via email.

Procedure

1. Navigate to **System Definition > Scheduled Jobs**.
2. Click **New**.
3. Select and click on Automate the generation and distribution of a report



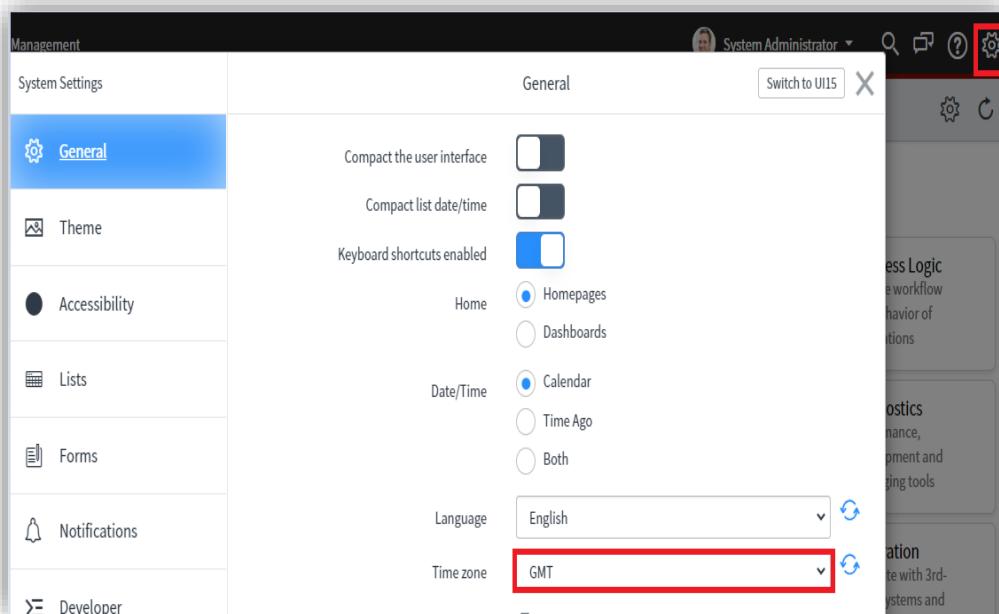
4. Fulfil scheduled jobs form

The screenshot shows the 'Scheduled Email of Report' configuration form. At the top, it displays the report name 'Category Of Incidents', application 'Global', and an active status. Below this, the 'Report' field is set to 'Category of Incidents'. The 'Run' schedule is set to 'Monthly' on the 15th at 04:34:00. The 'Type' is set to 'PDF-landscape'. The 'Email addresses' field contains 'srinivassnow2113@gmail.com'. The 'Users' section shows an email address 'srinivassnow2113@gmail.com' with a lock icon. The 'Groups' section shows a lock icon. There are also 'Conditional' and 'Omit if no records' checkboxes.

5. Click on Submit

Before testing this, we should set up particular time zone for this instance

1. Navigate to **Setting → Time Zone**
2. Select **GMT** time Zone



Create a new scheduled Job

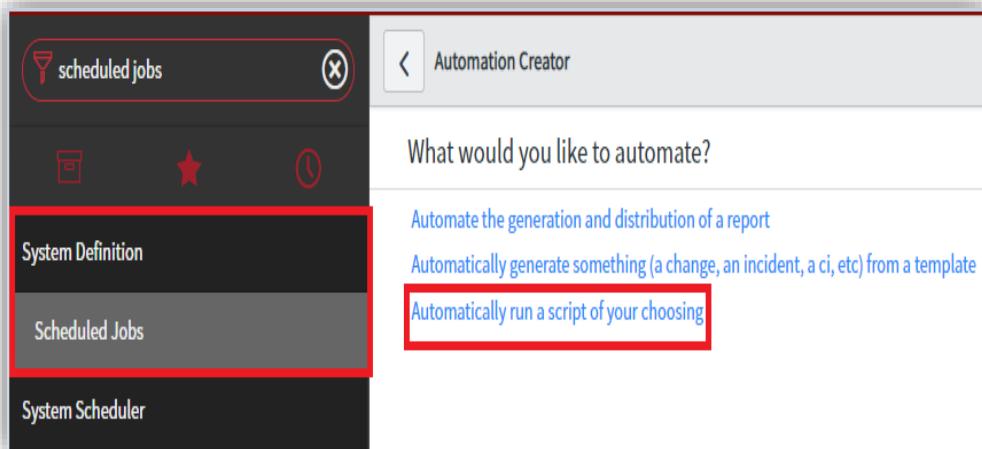
All resolved incidents should get closed automatically on **every 6 months**

Exercise: 2

All resolved incidents should get closed automatically on **every 6 months**

Procedure

1. Navigate to **System Definition > Scheduled Jobs**.
2. Click **New**.
3. Select and click on Automatically run a script of your choosing



4. Fulfil scheduled jobs form
5. **Name:** All resolved incidents should get closed automatically
6. **Active:** True

The screenshot shows the 'Scheduled Script Execution' form for 'Closing Tickets'. The form has fields for Name (set to 'All resolved incidents should get closed automatically'), Active (checkbox checked), Application (set to 'Global'), and Conditional (checkbox empty). There are also 'Update', 'Execute Now', and 'Delete' buttons at the top. A note at the bottom explains time zone options for scheduled jobs.

Name: All resolved incidents should get closed automatically

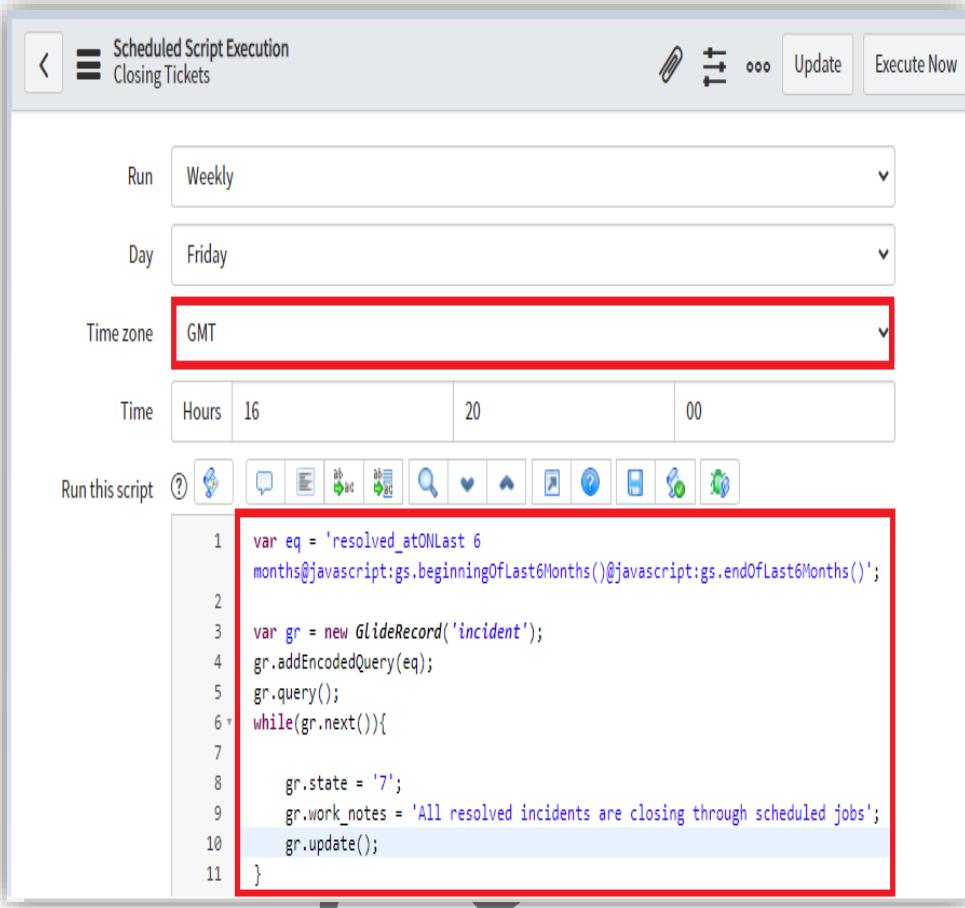
Active:

Application: Global

Conditional:

For scheduled job types that require an entered time, you have the option to enter an associated time zone. If no time zone is selected, the job will run at the entered time in time zone of the user who entered the time. If 'Use System Time Zone' is selected, the entered time will run in the time zone of the instance running the job.

7. **Run:** Weekly
8. **Friday:**
9. **Time Zone:** GMT
10. **Time:** 16:20:00
11. Click on **Submit**



12. Run this Script

```
var eq ='resolved_atONLast 6
months@javascript:gs.beginningOfLast6Months()@javascript:gs.endOfLast
6Months();'

var gr = new GlideRecord('incident');
gr.addEncodedQuery(eq);
gr.query();
while(gr.next()){

    gr.state = '7';
    gr.work_notes = 'All resolved incidents are closing through scheduled
jobs';
    gr.update();
}
```

Create a new scheduled Job

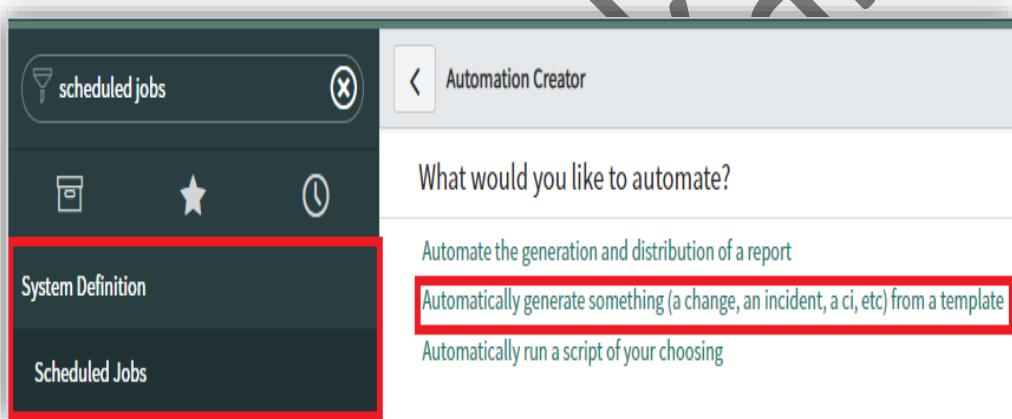
Create an incident ticket every week on Saturday in the mng 9AM

Exercise: 3

Create an incident ticket every week on Saturday in the mng 9AM

Procedure

1. Navigate to **System Definition > Scheduled Jobs**.
2. Click **New**.
3. Select and click on **Automatically generate something (a change, an incident, a ci, etc) from a template**



4. Create an incident ticket every week on Saturday in the mng 9AM
5. **Active:** True
6. **Run:** Weekly
7. **Time Zone:** GMT
8. **Time Hours:** 14:20:00
9. **Generate This:** Network Servers Maintanance

Scheduled Entity Generation
Create an incident ticket every week on Saturday in the mng 9AM

Name: Create an incident ticket every week on Saturday in the mng 9AM

Active:

Application: Global

Conditional:

For scheduled job types that require an entered time, you have the option to enter an associated time zone. If no time zone is selected, the job will run at the time zone of the user who entered the time. If 'Use System Time Zone' is selected, the entered time will run in the time zone of the instance running the job.

Run: Weekly

Day: Saturday

Time zone: GMT

Time: Hours: 14 Minutes: 20 Seconds: 00

Generate this: Network Servers Maintenance

10. Create Template

Template
Network Servers Maintenance

Name: Network Servers Maintenance

Table: Incident [incident]

Application: Global

User: System Administrator

Active:

Groups:

Global:

Short description: Network Server Maintanance

Template:

Short description	Network Server Maintenance
Category	Network
Subcategory	DNS
Caller	ITIL User

- 11. Name:** Network Servers maintenance
- 12. Table:** Incident
- 13. Active:** True
- 14. Short description:** Network Server Maintanance
- 15. Category:** Network
- 16. Subcategory:** DNS
- 17. Caller:** ITIL User

Srinivas Sunkara

Lesson-12

Business Rules

Agenda

- Business Rules Overview
- Advantages of Business rules
- What are actions performed with Business Rules
- Different Types of Business Rules
- Business Rules Process Flow
- How business rules will work
- Scripting In Business
- Practice with before and after business rules
- Practice with aync and display business rules
- Working with query business rule
- Business rules excercises
- Script Include Overview and Types of Script Includes
- Working with ,g_scratchpad, getRefference().
- Working with Glide Ajax
- Business Rules Interview Questions

Sri

Business Rules Overview

1. Business rules are the server side script which means that it will execute on server or database.
2. Business rules are the server side scripts logic that runs when record is displayed, inserted, updated or deleted
3. We can use BR to complete tasks like update the values on a form automatically when specific condition is met.
4. The table of BR is **sys_script** where all BR records are stored.
5. what all operations to be performed on database.
6. Create events for email notifications and script actions.
7. Business rule runs faster than other script in Service Now.
8. We can use business rules prevent unpredictable results and avoiding performance issues.



Business rules can make use of scripts to take actions on records in the database.

Advantages of Business rules

- Performance. When running code on the server, it often has a faster load time and processing time than a client script
- Business rules are not affected by type of browser which you are using
- Business rules will perform complex database lookups and assignment rules
- We can dot-walk many levels; however, three levels are often a recommend maximum
- Business Rules do NOT monitor forms. The forms shown in the graphics on this page represent a user interacting with the database by loading and saving records in a form.

What are actions performed with Business Rules

- ✓ Update record fields when a database query runs
- ✓ Modifying date formats
- ✓ Set field values on related records when a record is saved
- ✓ Manage failed log in attempts
- ✓ Generating events
- ✓ Determine if a user has a specific role
- ✓ Send emails to customers
- ✓ Generate and respond to events
- ✓ Display information message when record is inserted
- ✓ Compare two dates to determine which comes first chronologically
- ✓ Determine if today is a weekend or weekday
- ✓ Calculate the date when the next quarter starts
- ✓ Initiate integration and API calls to other systems
- ✓ Invoking web services and send REST messages and retrieve results
- ✓ Changing field values and Writing log messages
- ✓ If user is providing any invalid data may be that something is not as per the validation
- ✓ we want to put then user should not be able to get update or insert record.
- ✓ When we are inserting any record if we want to display a message on the top to user on the form then we can use BR as well.

Different Types of Business Rules

There are 4 types of business rule in Service Now as mentioned below. We selected business rule type from when dropdown available in when to run section. Below is the screenshot of business rule in Service Now.

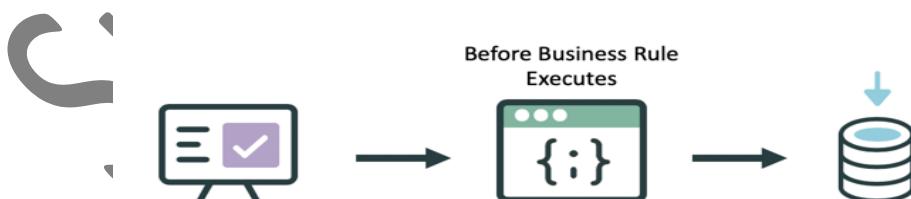
- ✓ Before
- ✓ After
- ✓ Async
- ✓ Display

Business Rule
New record

Name	My Test Business Rule	Application	Global																				
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>																				
Advanced <input checked="" type="checkbox"/>																							
<p>When to run</p> <ul style="list-style-type: none"> Actions Advanced <p>Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions.</p> <table border="1"> <tr> <td>When</td> <td>before</td> <td>Insert</td> <td><input type="checkbox"/></td> </tr> <tr> <td>Order</td> <td>before</td> <td>Update</td> <td><input type="checkbox"/></td> </tr> <tr> <td></td> <td>after</td> <td>Delete</td> <td><input type="checkbox"/></td> </tr> <tr> <td></td> <td>async</td> <td>Query</td> <td><input type="checkbox"/></td> </tr> <tr> <td></td> <td>display</td> <td></td> <td></td> </tr> </table>				When	before	Insert	<input type="checkbox"/>	Order	before	Update	<input type="checkbox"/>		after	Delete	<input type="checkbox"/>		async	Query	<input type="checkbox"/>		display		
When	before	Insert	<input type="checkbox"/>																				
Order	before	Update	<input type="checkbox"/>																				
	after	Delete	<input type="checkbox"/>																				
	async	Query	<input type="checkbox"/>																				
	display																						

Before Business Rule

- ✓ Code written in before business rule get executed when user submits the form and data is not saved in database.
- ✓ Before Business Rules execute their logic before a database operation occurs.
- ✓ Use before Business Rules when field values on a record need to be modified before the database access occurs.
- ✓ Before Business Rules run before the database operation so no extra operations are required.
- ✓ For example, concatenate two fields values and write the concatenated values to the Description field.



After Business Rule

- ✓ Use to update information on related objects that need to be displayed immediately,
- ✓ After Business rules are executed synchronously.
- ✓ After Business Rules execute their logic **immediately** after a database operation occurs and before the resulting form is rendered for the user.
- ✓ Use after Business Rules when no changes are needed to the record being accessed in the database.
- ✓ For example, use an after Business Rule when updates need to be made to a record related to the record accessed.
- ✓ If a record has child records use an after Business Rules to propagate a change from the parent record to the children.



Async Business Rule

Sync business rules are used to update information on related objects that do not need to be displayed immediately,

Use sync Business Rules instead of after Business Rules whenever possible to benefit from executing on the scheduler thread.

- ✓ Like after Business Rules, sync Business Rules execute their logic after a database operation occurs.
- ✓ Unlike after Business Rules, sync Business Rules execute asynchronously.
- ✓ Sync Business Rules execute on a different processing thread than before or after Business Rules.
- ✓ They are queued by a scheduler to be run as soon as possible.
- ✓ This allows the current transaction to complete without waiting for

the Business Rules execution to finish and prevents freezing a user's screen.
Use Async Business Rules when the logic can be executed in near real-time as opposed to real-time (after Business Rules).



Display Business Rule

Use to provide client-side scripts access to server-side data.

display Business rule runs every time the form is displayed and the form attempts to save due to `current.update ()`. User might not have filled out the form all the way and it is an annoying experience to the user.

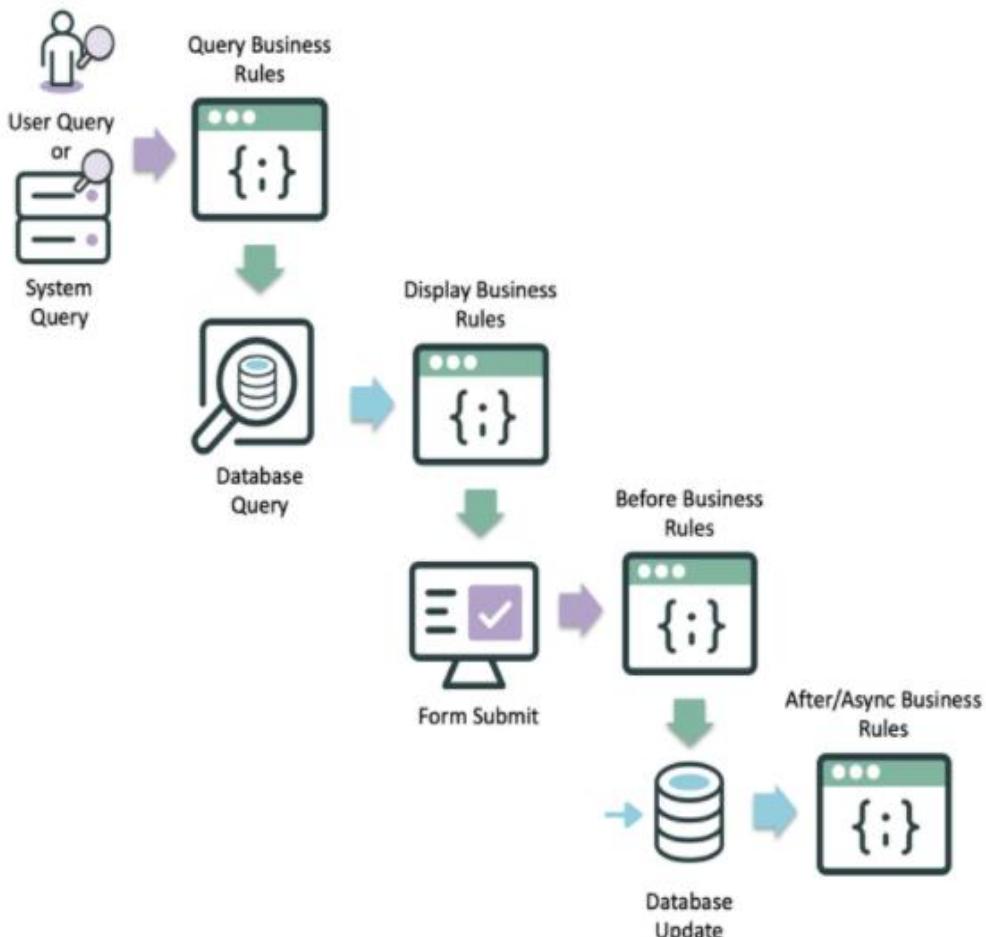
 Don't use `current.update ()` in a business rule! There are certain situations when it is ok, but very rarely. Same goes with using `g_form.save ()` in a client script.

- ✓ Display Business Rules execute their logic when a form loads and a record is loaded from the database.
- ✓ They must complete execution before control of the form is given to a user.
- ✓ The purpose of a display Business Rule is to populate an automatically instantiated object, `g_scratchpad`.
- ✓ The `g_scratchpad` object is passed from the display Business Rule to the client-side for use by client-side scripts.
- ✓ Recall that when scripting on the client-side, scripts only have access to fields and field values for fields on the form and not all of the fields from the database.
- ✓ Use the `g_scratchpad` object to pass data to the client-side without modifying the form. The `g_scratchpad` object has no default properties.



Business Rule Process Flow

A table can have multiple Business Rules of different When types. The order in which the Business Rules execute first



How business rules will work

If you want to configure new business rules, you first need to determine when the business rule should run and what action it should take.

When business rules run

Business rules are running based on two sets of criteria:

1. When to run the business rule in which we configure the time when BR is triggered.
2. In this we need to define timing and operation to trigger the BR.
3. So, one type of BR we mention the time when BR is configured for the record being modified or accessed.

When business rules are running based on action

Action in which we mention action we want to take when BR I triggered which can do via configuration or script.

This is the type of data base operation when both are met with the condition mentioned in the BR. Then BR is triggered.

We have 4 Database operations

- Insert
- Update
- Query
- Delete

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions:

When	before
Order	100

Insert	<input type="checkbox"/>
Update	<input checked="" type="checkbox"/>
Delete	<input type="checkbox"/>
Query	<input type="checkbox"/>

Filter Conditions Add Filter Condition Add "OR" Clause

Prevent Recursive Business Rules

Do not use **current.update()** in a Business Rule script. The **update()** method triggers Business Rules to run on the same table for insert and update operations, potentially leading to a Business Rule calling itself over and over. Changes made in before Business Rules are automatically saved when all before Business Rules are complete, and after Business Rules are best used for updating related, not current, objects. When a recursive Business Rule is detected, ServiceNow stops it and logs the error in the system log. However, this behavior may cause system performance issues and is never necessary.

You can prevent recursive Business Rules by using the **setWorkflow ()** method with the false parameter, **current.setWorkflow(false)**. This will stop Business Rules and other related functions from running on this database access. The combination of the **update ()** and **setWorkflow()** methods is only recommended in special circumstances where the normal

before and after guidelines mentioned above do not meet your requirements

There are four Global Variables available in Business Rules

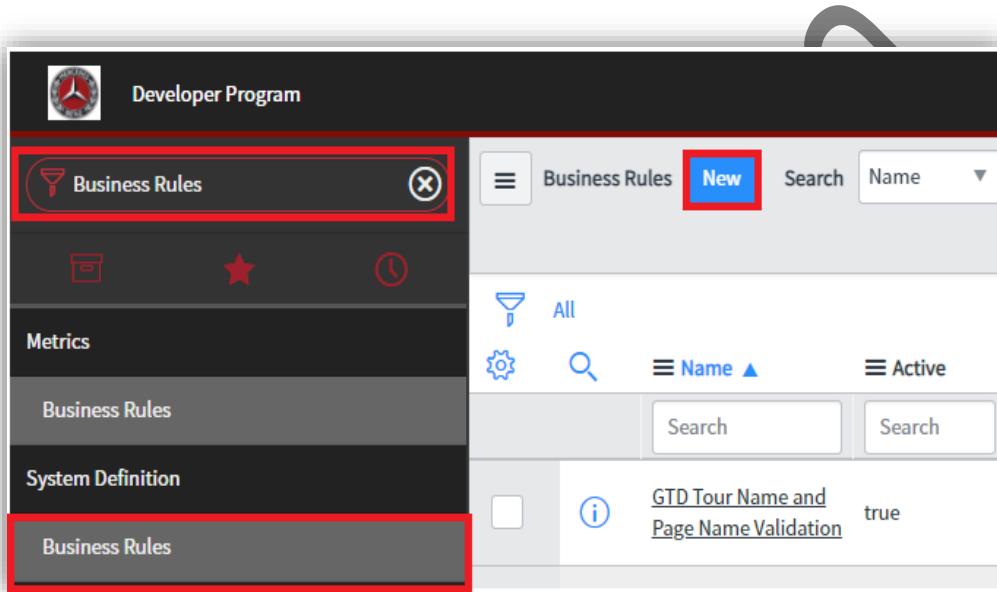
- ✓ **Current** – this is something in which basically stores state of the same record we are working on.
- ✓ **Previous** – which basically captures the previous state of the record we are working on before any update or delete. For example, if there is any change in the value for any particular field that's something we can access it with current and previous as well. Previous is the basically previous of the record.
- ✓ **g_scratchpad** – this is which we can access in BR. The scratchpad is available on display type of BR that means type is display. It is used to pass the information to the client.
- ✓ **Glide System** – which is server side script which where we have different methods and glide system methods as well.

Exercise: 1

- 1. Condition:** When you are selected the category is **Network** and subcategory is **VPN** from Incident table
- 2. Action:** Assignment group should be **Network**

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**



3. Fill the business rule Form
4. **Name:** Auto Assign to Assignment Group
5. **Table:** Incident
6. **Active:** True

When to run:

7. **Insert:** True
8. **Update:** True
9. **Category:** Network
10. **Subcategory:** VPN
11. Click on **Save**

Business Rule
New record

Name	Auto Assign to Assignment Group	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
		Advanced	<input type="checkbox"/>
When to run		Actions	
Insert <input checked="" type="checkbox"/> Update <input checked="" type="checkbox"/>			
Filter Conditions		Add Filter Condition	Add "OR" Clause
All of these conditions must be met			
Category is Network Subcategory is VPN			

Action:

Set Field values: Assignment Group → To → Network

Add message: This ticket has been assigned to Network assignment group

When to run **Actions**

Specify field values using the Set field values choice lists:

- To: a value determined by the options available for that field.
- Same as: a value taken from another field.
- To (dynamic): A value relative to the user configuring the business rule, or a user with a specific role.

Set field values

Assignment group	To	Network
-- choose field --	To	-- value --

Add message

Message

The ticket has been assigned to Network Assignment Group

Scripting in Business Rules

- We can also perform scripting in business rule for the functionality which we are not able to achieve with just configuration.
- Business Rules scripts use the server-side APIs to take actions. Those actions could be, but are not limited to
 - ✓ Invoking web services
 - ✓ Changing field values
 - ✓ Modifying date formats
 - ✓ Generating events
 - ✓ Writing log messages
 - ✓ Displaying Information and Error messages

Basically we use the “Advanced” option and we can write server side script in this script section ideally there is no spatial syntax for BR however they are global variables which we can use to access of data records in our script.

Create Business rule with script

Exercise: 2

1. **Condition:** When the user selected category is **Network** from incident table
2. **Action:** Assignment group should be **Network Assignment Group**

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**

The screenshot shows the ServiceNow developer program interface. The top navigation bar includes a logo and the text "Developer Program". Below the header, a search bar contains the text "business rule". The main content area has a title "Business Rules" with a "New" button highlighted by a red box. To the right of the title are "Search" and "Name" dropdown filters. On the left, a sidebar lists categories: Metrics, Business Rules, System Definition, Business Rules (which is selected and highlighted with a red box), System Diagnostics, and Session Debug. The main panel displays three items in a list:

- (i) [GTD Tour Name and Page Name Validation](#)
- (i) [Able to disable parameterized testing](#)
- (i) [Abort action if no license type](#)

3. **Name** : Auto assign to Group
4. **Table** : Incident
5. **Advanced**: True
6. **Insert**: True
7. **Update**: true

Srinivas

Name: Auto Assign to Group Application: Global

Table: Incident [incident] Active:

Advanced:

When to run: Actions Advanced

When: before Order: 100

Actions: Insert Update Delete Query

Filter Conditions: Add Filter Condition Add "OR" Clause

Condition: Category is Network

8. Condition: Category is Network

9. Click on Submit

Advanced

Condition:

Script:

```

1  (function executeRule(current, previous /*null when async*/) {
2
3    current.assignment_group = '287ebd7da9fe198100f92cc8d1d2154e';
4
5  })(current, previous);

```

10. Check Result

Before Business Rules

Working with Before Business Rule

Before Business Rules execute their logic before a database operation occurs.

Use before Business Rules when field values on a record need to be modified before the database access occurs.

Major Use Cases of Before Business Rule:

- Use business rules to accomplish tasks like automatically changing values in form fields when certain conditions are met.
- Create events for email notifications and script actions.
- Abort update of record if form does not have expected value.
- Auto Populate the field value while creating of incident record.
- User should not delete a record if condition does not match.
- Date should be updated before insert of a record.

Before Business Rules without Scripting

Exercise: 3

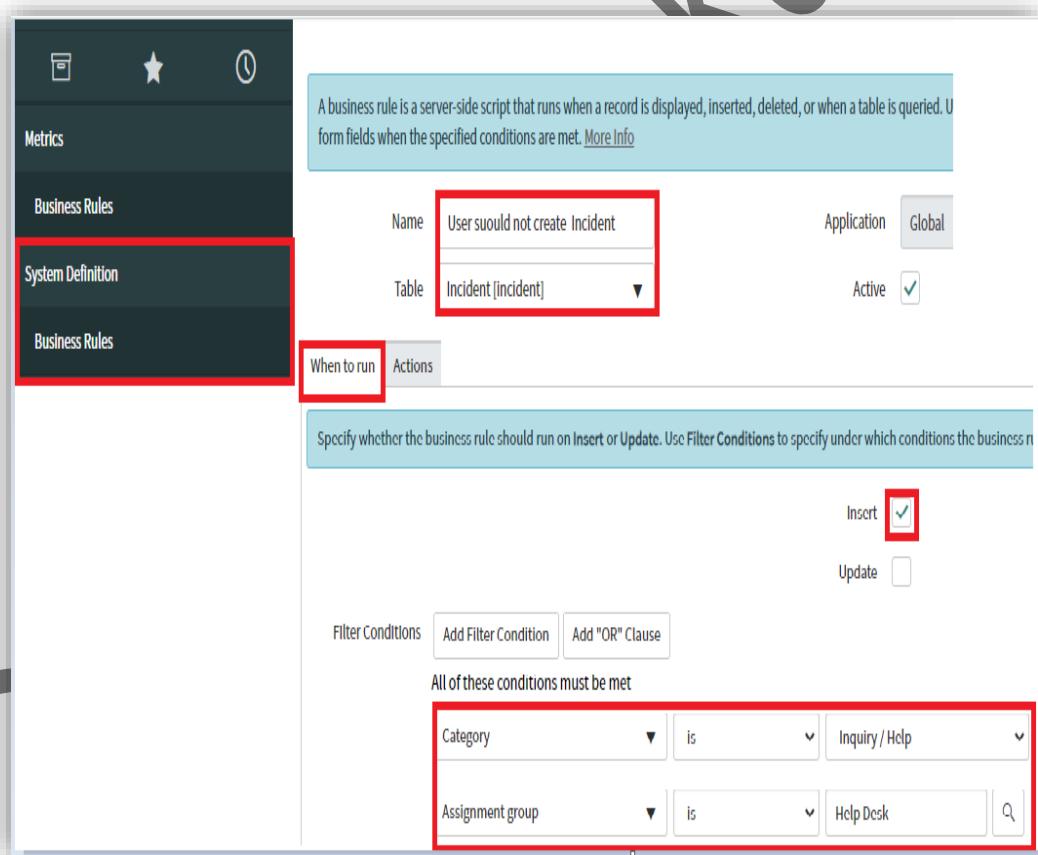
User should not be able to create new incident record, if category is 'Inquiry/Help' and assignment group is 'Service Desk'.

Action: Action should be **Abort** and display action **Message**

Message: ("User should not be able to create new incident record with above criteria")

Procedure

1. Navigate to **System Definition**→**Business Rules**
2. Click on **New**



3. Fill the business rule form
4. **Name:** User should not create Incident
5. **Table:** Incident
6. **Active:** True

When to Run

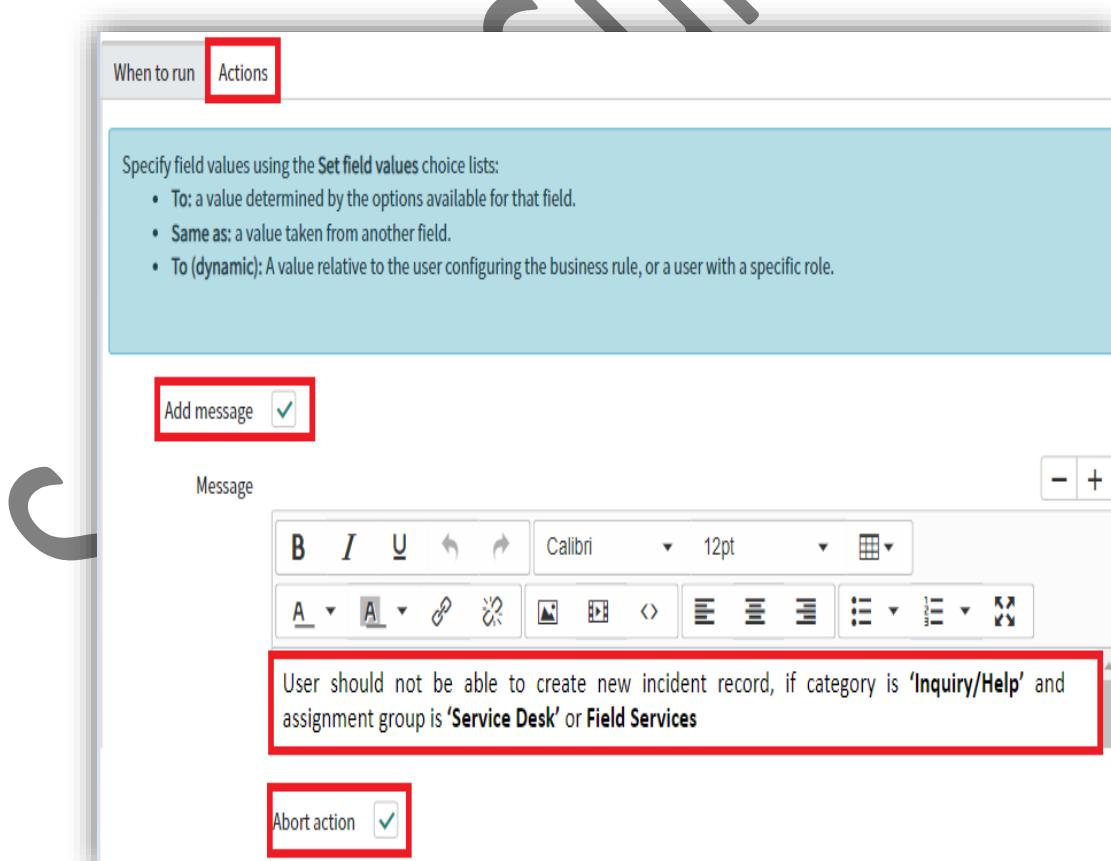
7. **When:** Before
8. **Order:** 300
9. **Insert:** Checked **true**

Filter Conditions

10. **Category** is Inquiry/Help
11. **Assignment Group** is Service Desk

Actions

12. Abort action: Checked **True**



Before Business Rule with Scripting

Exercise: 4

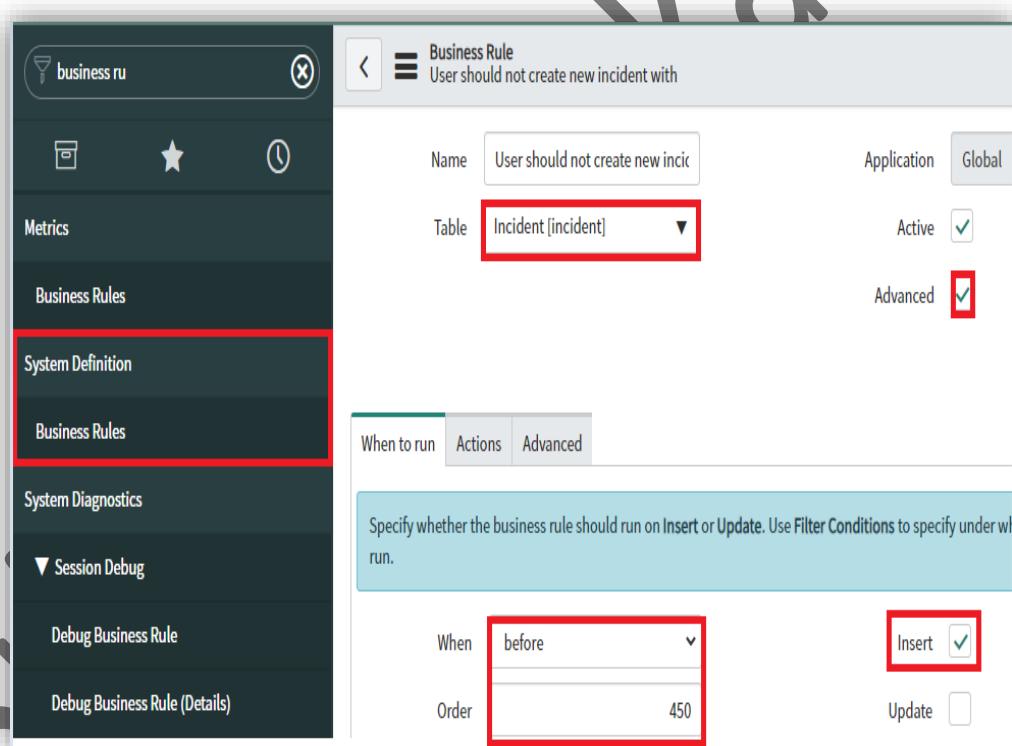
User should not be able to create new incident record, if there is an existing incident with same **Short Description**, **Category** and **Caller**.

Action: Action should be **Abort** and display action **Message**

Message : ('User should not create new incident with same details')

Procedure

1. Navigate to **System Definition**→**Business Rules**
2. Click on **New**



3. Fill the business rule form
4. **Name:** User should not create new Incident
5. **Table:** Incident
6. **Active:** True
7. **Advanced:** True

When to Run

8. When: Before
9. Order: 450
10. Insert: Checked true

Advanced

The screenshot shows the 'Business Rule' configuration window. The title bar says 'Business Rule' and 'User should not create new incident with'. The top navigation bar has tabs for 'When to run', 'Actions', and 'Advanced', with 'Advanced' being the active tab and highlighted with a red box. Below the tabs is a 'Condition' input field. Under the condition field is a 'Script' editor with a toolbar above it containing various icons. The script code is as follows:

```
1 (function executeRule(current, previous /*null when async*/ ) {  
2  
3     var gr = new GlideRecord('incident');  
4     gr.addQuery('short_description', current.short_description);  
5     gr.addQuery('caller_id', current.caller_id);  
6     gr.addQuery('category', current.category);  
7     gr.query();  
8  
9     if(gr.next()){  
10         gr.setAbortAction(true);  
11         gs.addErrorMessage('You can not create new incident record with same  
12             details');  
13     }  
14 })(current, previous);
```

Exercise: 5

User should not change the **Priority** once created an incident ticket

Action: Action should be **Abort** and display action **Message**

Message ('User should not change the priority once created an incident ticket')

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**

The screenshot shows a software interface for managing business rules. On the left, there's a sidebar with icons for Home, Metrics, Business Rules, System Definition (which is highlighted with a red box), and Business Rules. The main area is titled 'Business Rule' and shows a rule named 'User should not change priority'. A tooltip explains that a business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table automatically changes values in form fields when the specified conditions are met. The rule details are as follows:

Name	User should not change priority	Application	Global
Table	Incident [incident]	Active	<input checked="" type="checkbox"/>
		Advanced	<input checked="" type="checkbox"/>

3. Fill the business rule form
4. **Name:** User should not change the Priority once created an incident ticket
5. **Table:** Incident
6. **Active:** True

When to Run

7. **When:** Before
8. **Order:** 200
9. **Update:** true

Filter Conditions

10. Priority Changes

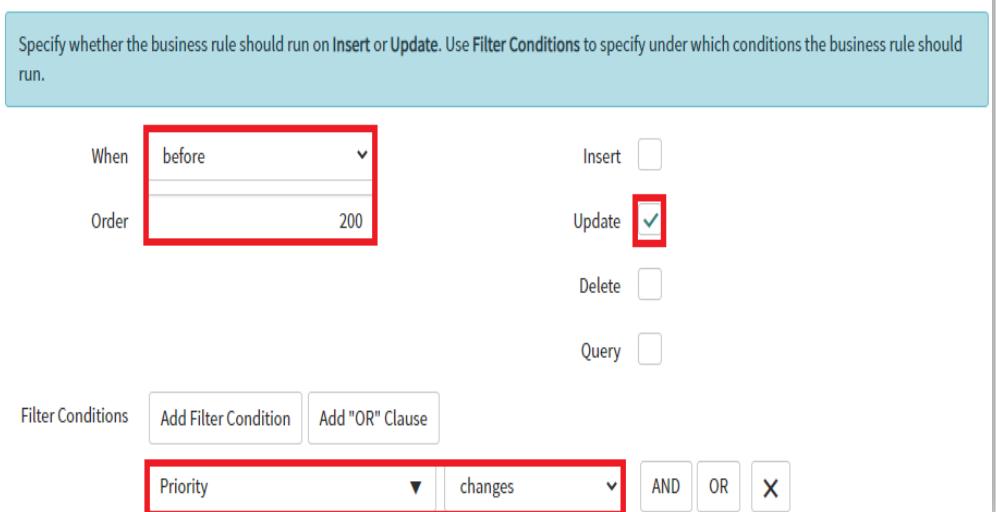
When to run Actions Advanced

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule should run.

When: before Insert
Order: 200 Update Delete Query

Filter Conditions: Add Filter Condition Add "OR" Clause

Priority changes AND OR X



Actions

11. **Add message:** true
12. **Message:** User should not change the Priority once created an incident ticket
13. **Abort action:** True

When to run Actions Advanced

Specify field values using the Set field values choice lists:

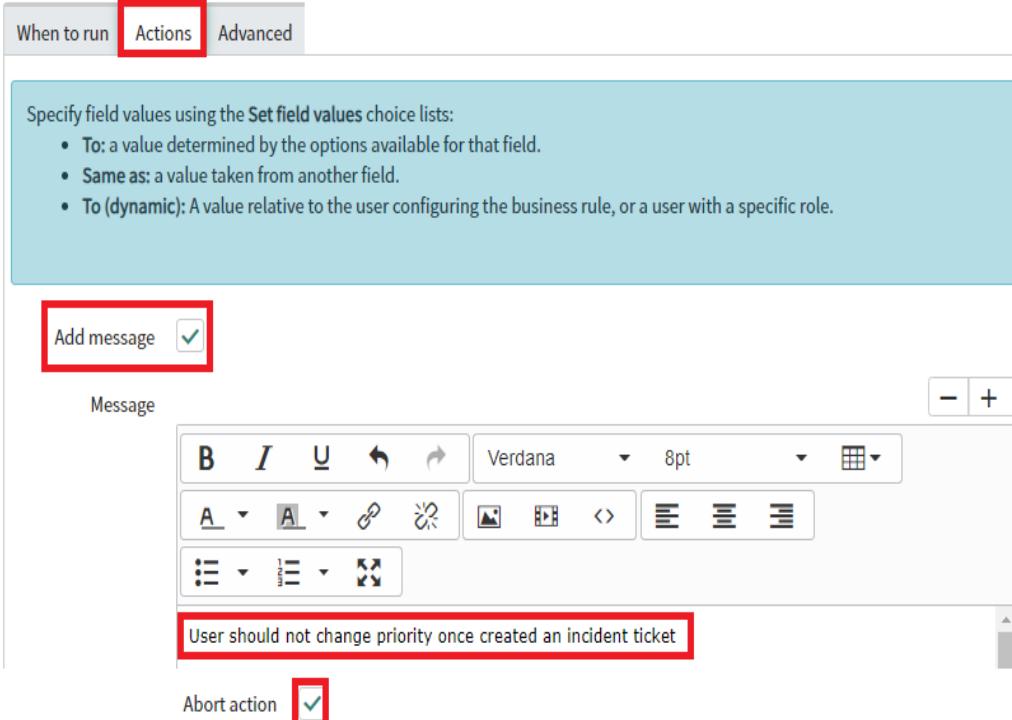
- To: a value determined by the options available for that field.
- Same as: a value taken from another field.
- To (dynamic): A value relative to the user configuring the business rule, or a user with a specific role.

Add message

Message

User should not change priority once created an incident ticket

Abort action



Exercise: 6

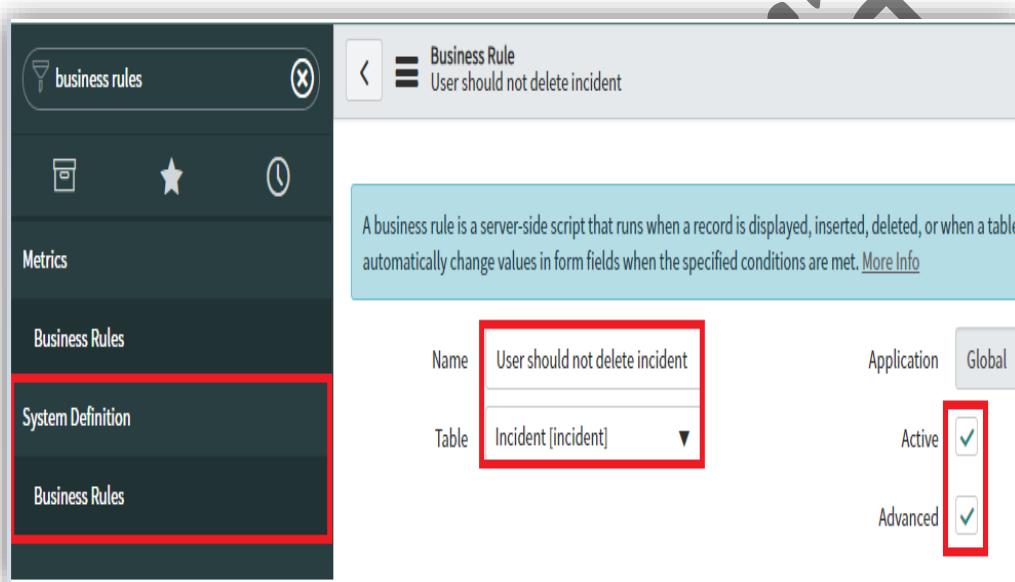
User should not delete incident ticket when incident state is **NEW**

Action: Action should be **Abort** and display action **Message**

Message :('User should not delete incident ticket in NEW State')

Procedure

1. Navigate to **System Definition**→**Business Rules**
2. Click on **New**



3. Fill the business rule form
4. **Name:** User should not delete Incident ticket
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

When to Run

8. **When:** Before
9. **Order:** 350
10. **Delete:** true

Filter Conditions

11. **State is New**

Business Rule
User should not delete incident

When to run Actions Advanced

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule should run.

When **before** Insert
Order **350** Update
Delete **Query**

Filter Conditions Add Filter Condition Add "OR" Clause

State is New

```
when before
order 350
delete
query
```

State is New

Advanced

Business Rule
User should not delete incident

When to run Actions Advanced

Condition

Script

```
(function executeRule(current, previous /*null when async*/ ) {  
    current.setAbortAction(true);  
    gs.addErrorMessage('User should not delete incident ticket when incident state is NEW');  
})(current, previous);
```

```
(function executeRule(current, previous /*null when async*/ ) {  
    current.setAbortAction(true);  
    gs.addErrorMessage('User should not delete incident ticket when incident state is NEW');  
})(current, previous);
```

After Business Rules

Working with After Business Rules

After business rules will get execute after the user submits the form and after any action is taken on the record in the database.

Major Use Cases of After Business Rule:

- Showing **Info** or **Error** Messages after creation or updating of record
- Create incident task after creation of Incident ticket
- Update the work notes of child incident when parent work notes are updated
- Create Knowledge article after close incident
- Delete all associated records after deletion of parent record

After Business Rules without Scripting

Exercise: 7

We should display **Info message** after creation of new Incident ticket.

Message ('New incident has been created')

Procedure

1. Navigate to **System Definition**→**Business Rules**
2. Click on **New**

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. It automatically changes values in form fields when the specified conditions are met. [More Info](#)

Name: Display info message on top of f...
Table: Incident [incident]
Application: Global
Active:
Advanced:

3. Fill the business rule form
4. **Name:** Display info message
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions it runs.

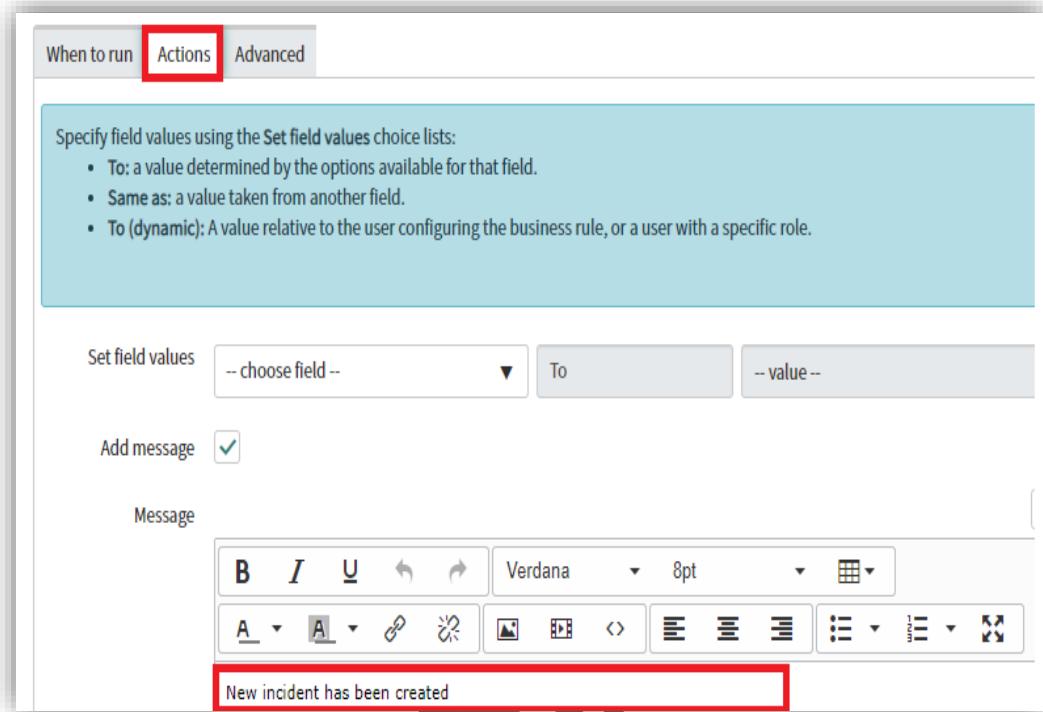
When: after
Order: 100

Insert:
Update:
Delete:
Query:

When to Run

1. **When:** After
2. **Order:** 100
3. **Insert:** true

Actions



After Business Rules with Scripting

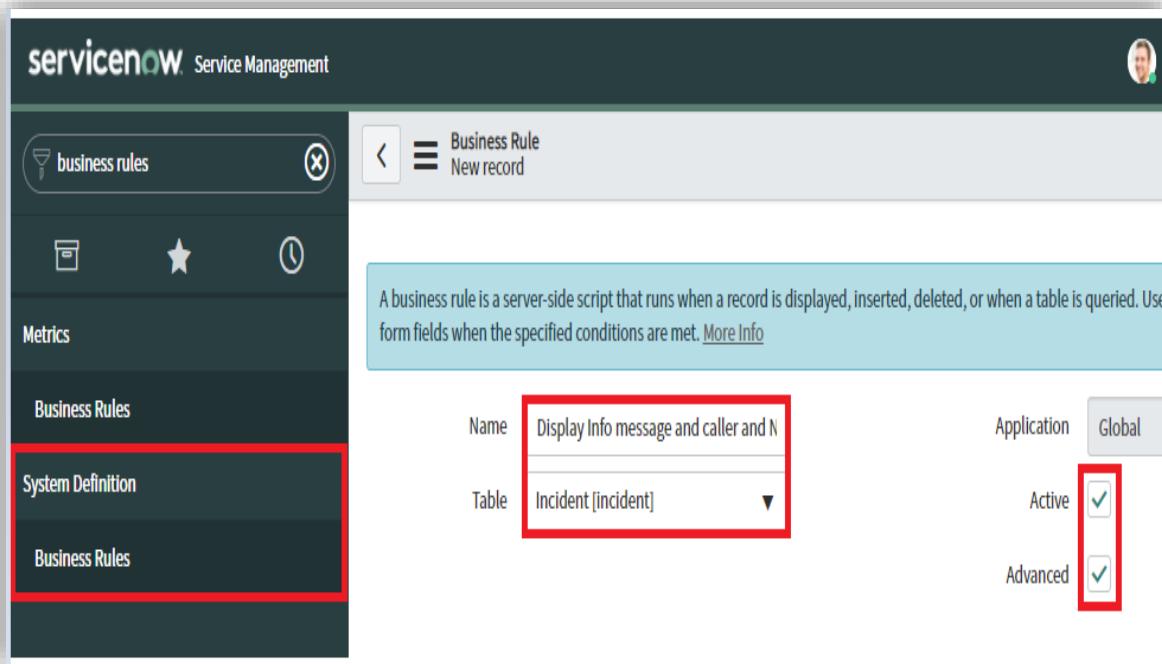
Exercise 8

We should display **Info message**, **Caller** and **Incident Number** after creation of new Incident ticket.

Message ('New incident has been created by caller and incident number')

Procedure

1. Navigate to **System Definition** → **Business Rules**
2. Click on **New**



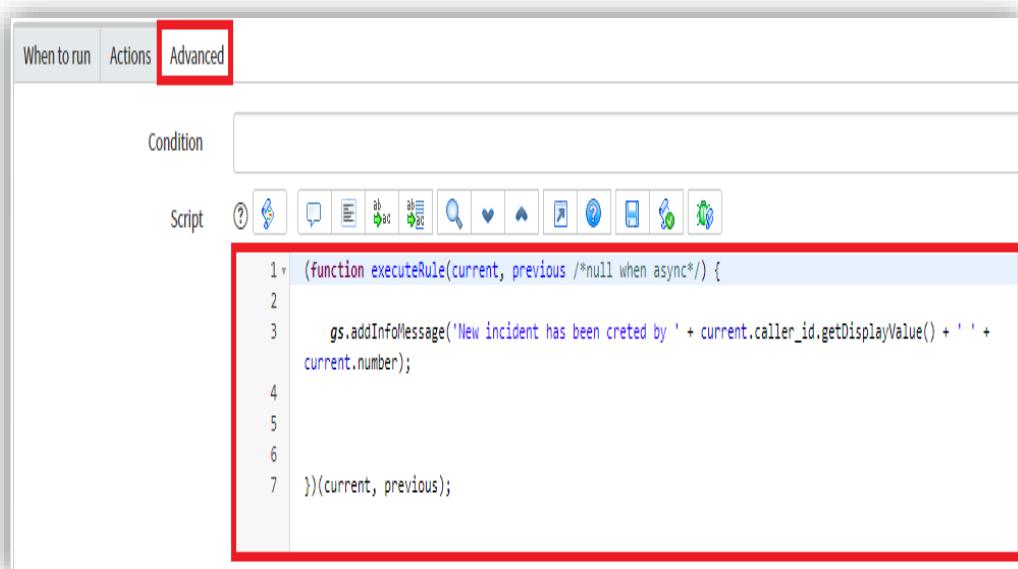
3. Fill the business rule form
4. **Name:** Display info message and Caller and Incident Number
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

This screenshot shows the 'When to run' tab of the business rule configuration. It includes tabs for 'When to run', 'Actions', and 'Advanced'. The 'When to run' tab contains a section for specifying when the rule should run: 'When' (set to 'after') and 'Order' (set to '200'). It also lists actions: 'Insert' (checked), 'Update' (unchecked), 'Delete' (unchecked), and 'Query' (unchecked). A large watermark 'ASSURE' is diagonally across the center of the screen.

When to Run

1. When: After
2. Order: 200
3. Insert: true

Advanced



```
(function executeRule(current, previous /*null when async*/) {  
    gs.addInfoMessage("New incident has been created by " + current.caller_id.getDisplayValue() + ' ' +  
    current.number);  
})(current, previous);
```

Exercise: 9

Worknotes should be update when you are changed assignment group value
And display info message about assignment group updating

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**

business rules

Business Rule
Assignment Group Change

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a automatically change values in form fields when the specified conditions are met. [More Info](#)

Name: Assignment Group Change

Table: Incident [incident]

Application: Global

Active:

Advanced:

3. Fill the business rule form
4. **Name:** Worknotes should be update on assignment group change
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

When to run Actions Advanced

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the rule runs.

When: after

Order: 300

Insert:

Update:

Delete:

Query:

Filter Conditions: Assignment group changes

Add Filter Condition Add "OR" Clause

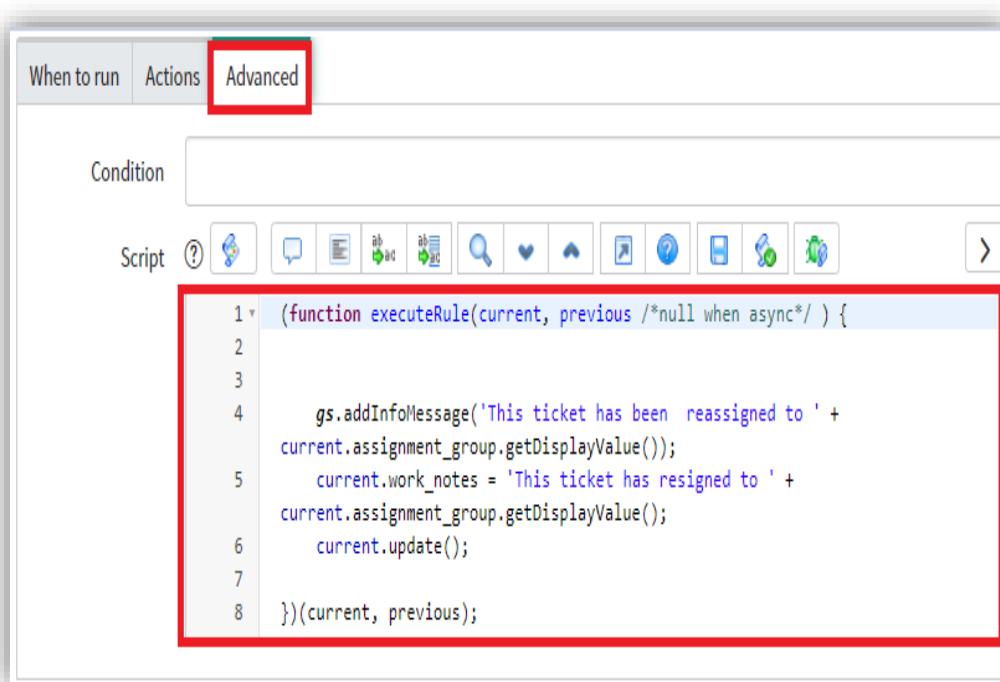
When to run

8. When: After
9. Update: True
10. Order:300

Filter Condition

Assignment Group Changes

Advanced



Exercise: 10

Condition

When incident state changes to **On Hold** and **On Hold Reason** is **Awaiting Change**

Immediately the system should be creating new change record in to **(change_request)** table

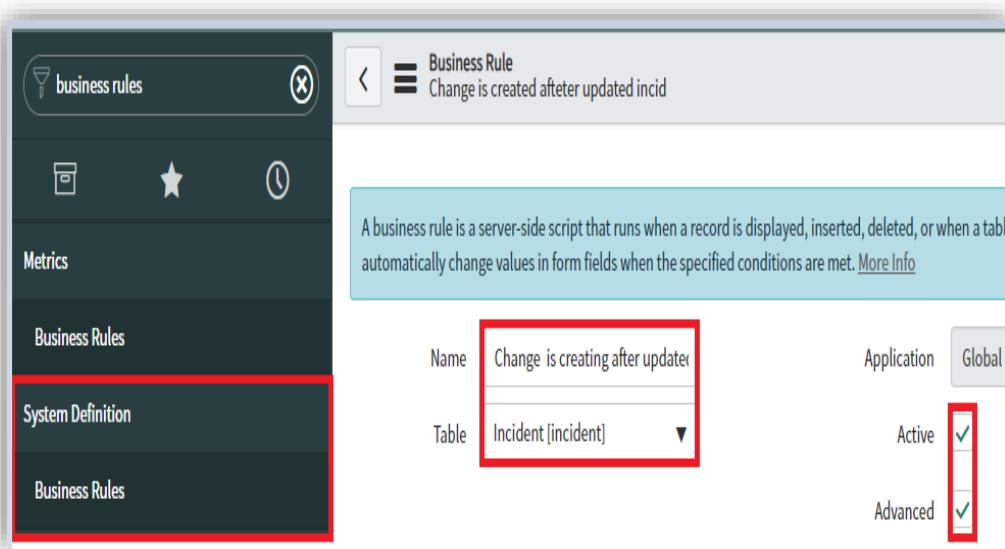
Action:

- Current logged in user set into **Requested By**
- Incident short description value set into **Change Short Description**
- Set change type is **Normal**
- Then display **Info Message** after created change record

Procedure

1. Navigate to **System Definition** → **Business Rules**

2. Click on **New**



3. Fill the business rule form

4. **Name:** Change is creating after updated incident with on hold state

5. **Table:** Incident

6. **Active:** true

7. **Advanced:** True

This screenshot shows the 'When to run' tab of the business rule configuration. It has tabs for 'When to run', 'Actions', and 'Advanced'. The 'When to run' tab is selected and highlighted with a red box. It contains fields for 'When' (set to 'after') and 'Order' (set to '100'). To the right, there are checkboxes for 'Insert', 'Update' (which is checked and highlighted with a red box), 'Delete', and 'Query'. Below these are buttons for 'Filter Conditions', 'Add Filter Condition', and 'Add "OR" Clause'. A note says 'All of these conditions must be met'. At the bottom, there are two rows of dropdown menus: 'State changes to On Hold' and 'On hold reason is Awaiting Change', both of which are highlighted with a red box.

When to run

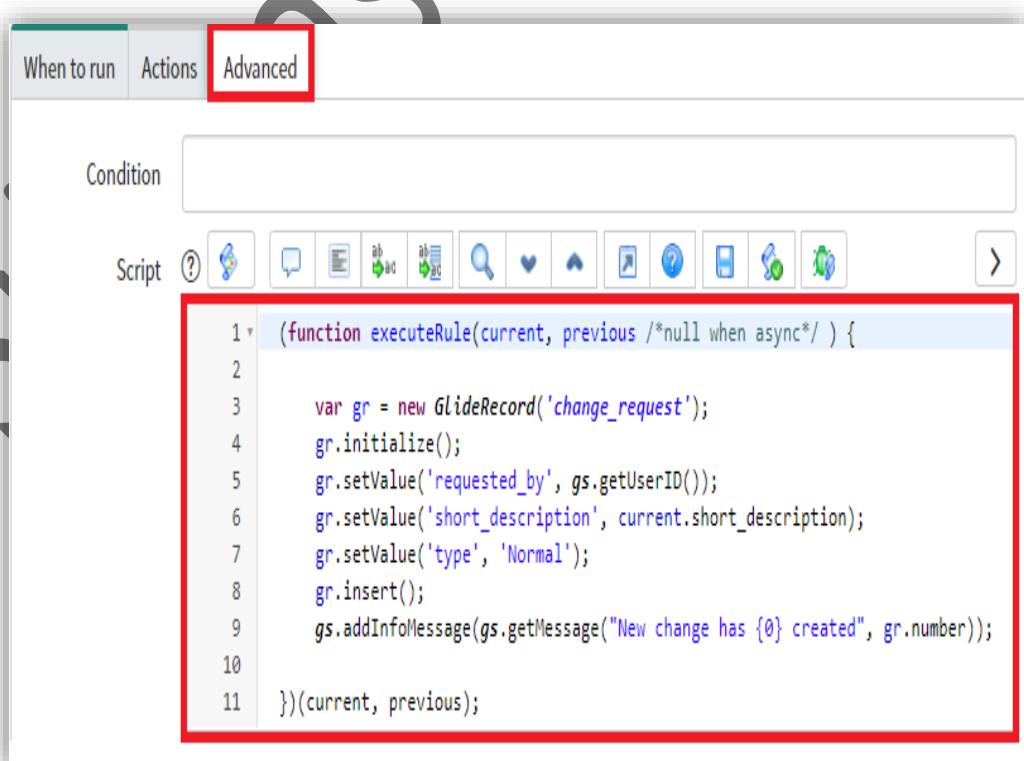
8. When: After
9. Order: 100
10. Update: True

Filter Condition

11. State changes to On Hold
12. On hold reason is Awaiting Change

Advanced

```
var gr = new GlideRecord('change_request');
gr.initialize();
gr.setValue('requested_by', gs.getUserID());
gr.setValue('short_description', current.short_description);
gr.setValue('type', 'Normal');
gr.insert();
gs.addInfoMessage(gs.getMessage("New change has {0} created",
gr.number));
```

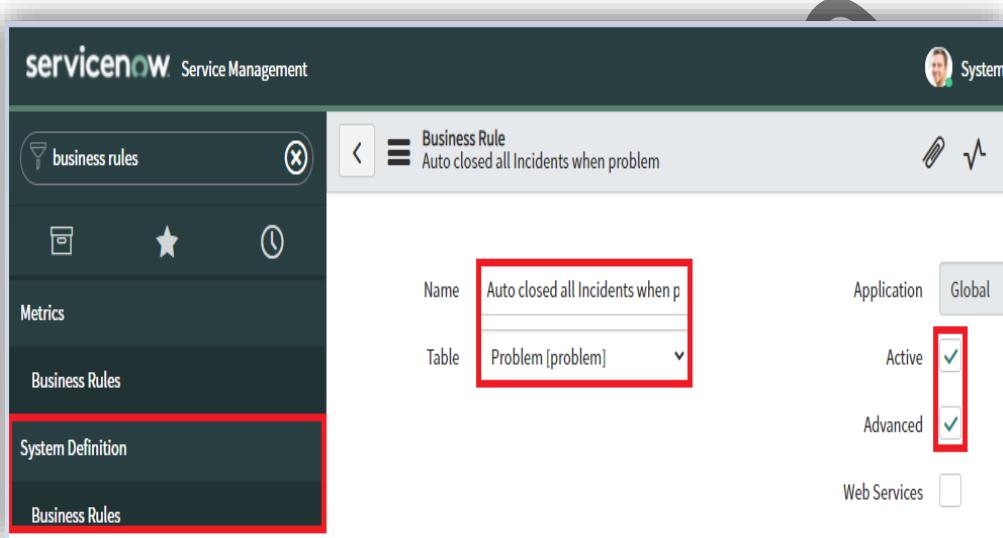


Exercise: 11

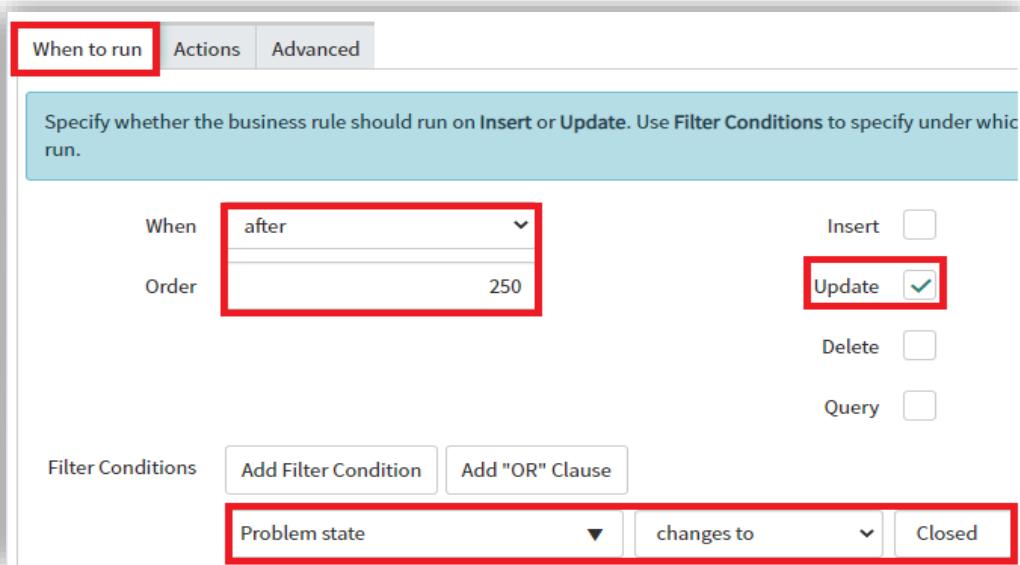
When we **Closed** problem ticket then automatically all related incidents should get closed

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**



3. Fill the business rule form
4. **Name:** Auto closed all incidents when problem is closed
5. **Table:** Problem
6. **Active:** true
7. **Advanced:** True



When to run

8. When: After
9. Order: 250
10. Update: True

Filter Condition

11. Problem State Changes to Closed

Advanced

```
var gr = new GlideRecord('incident');
gr.addQuery('problem_id', current.sys_id);
gr.query();
while (gr.next()) {

    gr.state = 7;
    gr.close_code = 'Solved (Work Around)';
    gr.close_notes = 'Successfully fixed issue';
    gr.setWorkflow(false); //don't want run other business rules
    gr.update();
}
```

The screenshot shows the 'Advanced' tab selected in the Business Rules editor. The 'Condition' field is empty. Below it is a toolbar with various icons, including a question mark, a script icon, and several utility icons. The main area contains a script editor with the following code:

```
1 (function executeRule(current, previous /*null when async*/){  
2  
3     var gr = new GlideRecord('incident');  
4     gr.addQuery('problem_id',current.sys_id);  
5     gr.query();  
6     while(gr.next()){  
7  
8         gr.state = 7;  
9         gr.close_code = 'Solved (Permanantly)';  
10        gr.close_notes = 'Sucessfully fixed issue';  
11        gr.setWorkflow(false); //don't want run other business rules  
12        gr.update();  
13    }  
14}  
15}  
16}  
17 })(current, previous);
```

Exercise: 12

Incident task should be creating automatically an incident is created and **Caller, Category, Priority, Short Description, Description, State** and **work notes** fields should be copied to Incident task from incident.

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**

servicenow Service Management

business rules

Metrics

Business Rules

System Definition

Business Rules

Business Rule
Incident task should be create automatic

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a automatically change values in form fields when the specified conditions are met. [More Info](#)

Name: Incident task should be create automatic

Table: Incident [incident]

Application: Global

Active:

Advanced:

3. Fill the business rule form
4. **Name:** Incident task should be creating automatically when new incident is created
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

When to run Actions Advanced

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which run.

When: after

Order: 350

Insert:

Update:

Delete:

Query:

When to run

1. When: After
2. Order: 350
3. Insert: True

Advanced

```
var gr = new GlideRecord('incident_task');
gr.initialize();
gr.incident = current.sys_id;
gr.priority = current.priority;
gr.assignment_group = current.assignment_group;
gr.assigned_to = current.assigned_to;
gr.cmdb_ci = current.cmdb_ci;
gr.work_notes = 'New incident task has created from ' + current.number;
gr.short_description = current.short_description;
gr.description = current.description;
gr.insert();
gs.addInfoMessage('Incident task has created #' + gr.number);
```

Advanced

The screenshot shows the 'Advanced' tab selected in a rule configuration interface. The tab bar includes 'When to run', 'Actions', and 'Advanced'. Below the tabs is a 'Condition' section with a search icon and several small icons for different actions. The main area is labeled 'Script' and contains the following code:

```
1 (function executeRule(current, previous /*null when async*/ ) {
2
3     var gr = new GlideRecord('incident_task');
4     gr.initialize();
5     gr.incident = current.sys_id;
6     gr.priority = current.priority;
7     gr.assignment_group = current.assignment_group;
8     gr.assigned_to = current.assigned_to;
9     gr.cmdb_ci = current.cmdb_ci;
10    gr.work_notes = 'New incident task has created from ' + current.number;
11    gr.short_description = current.short_description;
12    gr.description = current.description;
13    gr.insert();
14    gs.addInfoMessage('Incident task has created #' + gr.number);
15
16 })(current, previous);
```

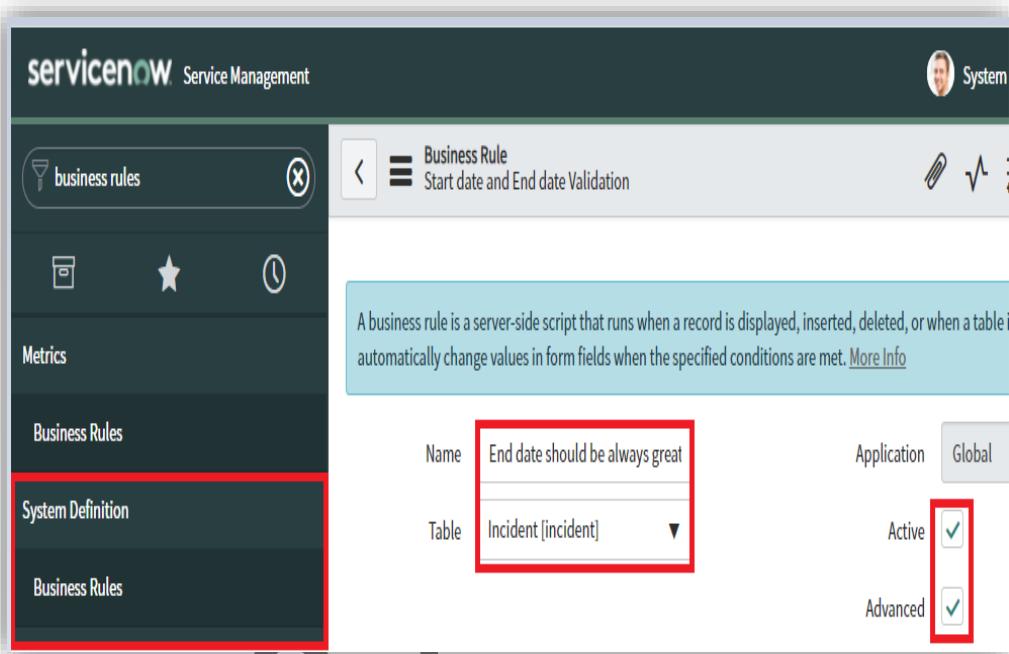
Exercise: 13

Date validation, End date should be always greater than start date

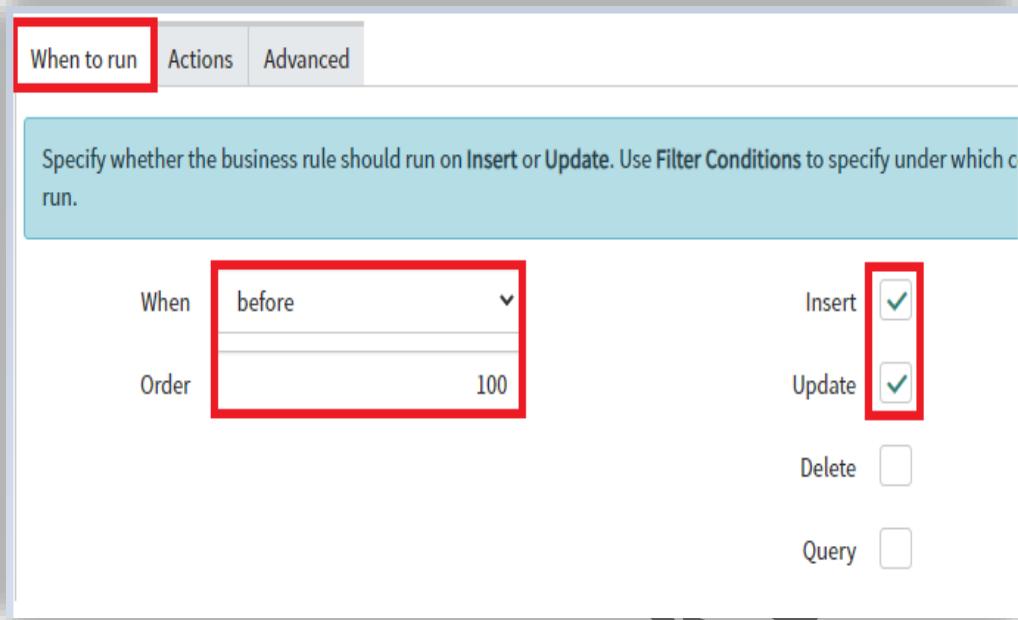
Need to create 2 custom fields (**Start Date** and **End Date**) in Incident table

Procedure

1. Navigate to **System Definition**→**Business Rules**
2. Click on **New**



3. Fill the business rule form
4. Name: End date should be always greater than start date
5. Table: Incident
6. Active: true
7. Advanced: True



When to run

8. When: before
9. Order: 100
10. Insert: True
11. Update: True

Advanced

```
var st_date = current.u_start_date.getGlideObject().getNumericValue();
var end_date = current.u_end_date.getGlideObject().getNumericValue();

if (st_date > end_date) {
    gs.addErrorMessage('Start date can not be greater than end date');
    current.setAbortAction(true);
```

When to run Actions **Advanced**

Condition

Script

```
1 (function executeRule(current, previous /*null when async*/ ) {  
2  
3     var st_date = current.u_start_date.getGlideObject().getNumericValue();  
4     var end_date = current.u_end_date.getGlideObject().getNumericValue();  
5  
6     if (st_date > end_date) {  
7  
8         gs.addErrorMessage('Start date can not be greater than end date');  
9         current.setAbortAction(true);  
10    }  
11}  
12  
13 })(current, previous);
```

Srinivas Sullu

Async Business Rules

Working with Async Business Rules

Async business rule run in background and are mostly used to handle things which are not required to be shown to the user after insert/update.

Async business rules are used to update information on related objects that do not need to be displayed immediately such as calculating metrics and SLAs. When the scheduler runs the scheduled job created from the business rule. The system creates a scheduled job from the business rule after the user submits the form and after any action is taken on the record in the database. Async Business Rule is triggered when user submits the form and any action is taken on the record in the database however it creates a schedule and run the job.

Major Use Cases of Async Business Rule:

- Send REST API call to another application on update of any ticket.
- Calculating SLAs and Metrix
- Create events through event registry
- Use to update information on the current object
- While importing data into Servicenow table, the new user record is inserted & notification is send to user's manager.
- Sending email notifications through events

Note: We cannot use “**changes ()**, **changesTo ()**, and **changesFrom ()**” methods in the scripting in Async Business Rules.

Note: Asynchronous business rules do not have access to the previous version of a record.

Difference between After and Async Business Rules

After BR	Async BR
After business rule is used when some fields needs to be updated and displayed to the user immediately after user saves the record.	Whenever async business rule runs it creates a scheduler for it in background and based on the availability of the nodes in ServiceNow it will execute.
After business rules are used to update information on related objects that need to be displayed immediately.	Async business rule is required when something is to be performed in Backend for updating some other table data etc and user need not know about that.
After the user submits the form and after any action is taken on the record in the database.	Mostly used to trigger API call from scoped app such as outbound SOAP/REST
After Business Rules are executed Synchronously	Async Business Rules are executed Asynchronously
We can access Previous and Current objects	We can't access previous object
User waits for response from and After Business Rules	When the scheduler runs the scheduled job created from the business rule. The system creates a scheduled job from the business rule
Create Knowledge article after close incident	We cannot use “ changes () , changesTo () , and changesFrom () ” methods in the scripting in Async Business Rules.
Update the work notes of child incident when parent work notes are updated	Calculating SLAs, Metrix and Sending email notifications through events

Async Business Rule with Scripting:

We should create below objects to work with **Async Business Rule**

About Task

When we created new incident with category of network then immediately send notification to assignment group

Working With

1. Event Registry
2. Notification
3. Business Rule

Exercise: 14

Create new event registry

Step 1: We need to create an event registry

Procedure

1. Navigate to System Policy → Events → Registry
2. Click on New

The screenshot shows the ServiceNow interface for creating a new Event Registry. The left sidebar is collapsed, and the main area displays the 'Event Registration' form under 'New record'. The form fields are as follows:

- Event name:** incident.critical.network (highlighted with a red box)
- Table:** Incident [incident] (highlighted with a red box)
- Fired by:** Business Rules (highlighted with a red box)
- Description:** When we created critical incident with category network then immediatly send notification to ass grp (highlighted with a red box)

3. Fill the event registry form

4. **Event Name:** incident.critical.network
5. **Table:** Incident
6. **Fired by:** Business Rules
7. **Description:** When we created new incident with category of network then immediately send notification to assignment group
8. Click on **Submit**

Create new notification

Step 2: We need to create a new notification

Procedure

1. Navigate to **System Notification** → **Email** → **Notification**
2. Click on **New**

Name	Critical incident with network	Application	Global
* Table	Incident [incident]	Active <input checked="" type="checkbox"/>	
* Category	Uncategorized	Allow Digest <input type="checkbox"/>	

3. Fill the **Notification** form
4. **Name:** Critical incident with network
5. **Table:** Incident
6. **Category:** Uncategorized
7. **Active:** True

When to send

8. **Send when:** Event is fired
9. **Event Name:** incident.critical.network

When to send **Who will receive** **What it will contain**

Notifications can be sent (if the specified Conditions are met) under one of the following circumstances:

- A record is **Inserted** or **Updated** into the **Table** specified above
- The specified event is fired
- Via a Flow Action

Send when: Event is fired

Event name: incident.critical.network

Conditions: Add Filter Condition | Add "OR" Clause

-- choose field -- ▾ -- oper -- -- value --

Who will receive

10. **Users:** Provide your personal mail ID
11. **Event parm 1 contains recipients:** True
12. **Exclude delegates:** True
13. **Send event creator:** True

When to send **Who will receive** **What it will contain**

Notifications can be sent to specific **Users** and **Groups** or to **Users/Groups** in fields on the record that generated this notification.

The Notification will not be sent to the User who caused the Notification to be triggered unless **Send to event creator** is checked.

The Notification will also be sent to Delegates of users unless **Exclude Delegates** is checked.

Users <input type="button" value="lock"/> <input type="button" value="add"/> <input type="text" value="srinivassnow2110@gmail.com"/>	Groups <input type="button" value="lock"/> <input checked="" type="checkbox"/>
Exclude delegates <input checked="" type="checkbox"/>	
Send to event creator <input checked="" type="checkbox"/>	
Event parm 2 contains recipient <input type="checkbox"/>	
Subscribable <input type="checkbox"/>	

What it will contain

14. **Subject:** New incident has been created by Caller: \${caller_id} and Category: \${category}
15. **Message HTML:** Caller:\${caller_id}, Category:\${category}

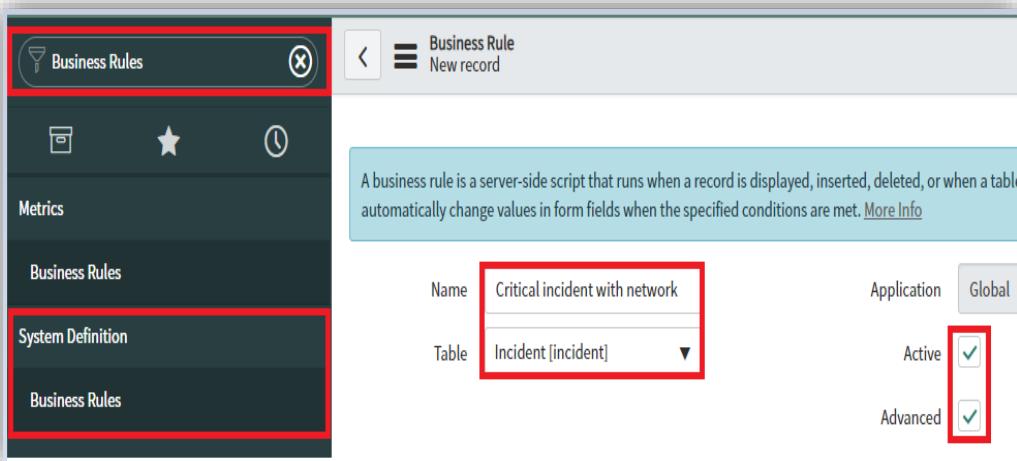
The screenshot shows a configuration interface for an email template. At the top, there are three tabs: 'When to send', 'Who will receive', and 'What it will contain'. The 'What it will contain' tab is selected and highlighted with a red border. Below the tabs, a note states: 'If using an Email Template then Subject and Message will be used from the template unless overridden with a Subject and/or Message in the rule.' The 'Email template' section contains a link to 'Unsubscribe and Preferences' and a search icon. The 'Subject' field contains the placeholder 'New incident has been created by Caller: \${caller_id} and Category: \${category}' with a red border around it. The 'Message HTML' section includes a rich text editor toolbar with bold (B), italic (I), underline (U), and other styling options. The message body contains the variables 'Caller: \${caller_id}' and 'Category: \${category}' with a red border around them. To the right of the message body is a 'Select variables:' panel listing various business objects and their properties, such as Business duration, resolve time, Caller, Category, Cause, and Change.

Create new Business Rule

Step 3: We need to create a new Business Rule

Procedure

1. Navigate to System Definition → Business Rules
2. Click on New



3. Fill the business rule form
4. **Name:** Critical incident with network
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule runs.

When	async	Order	100	Insert	<input checked="" type="checkbox"/>
				Update	<input type="checkbox"/>
				Delete	<input type="checkbox"/>
				Query	<input type="checkbox"/>

All of these conditions must be met

Priority	is	1 - Critical
Category	is	Network

When to run

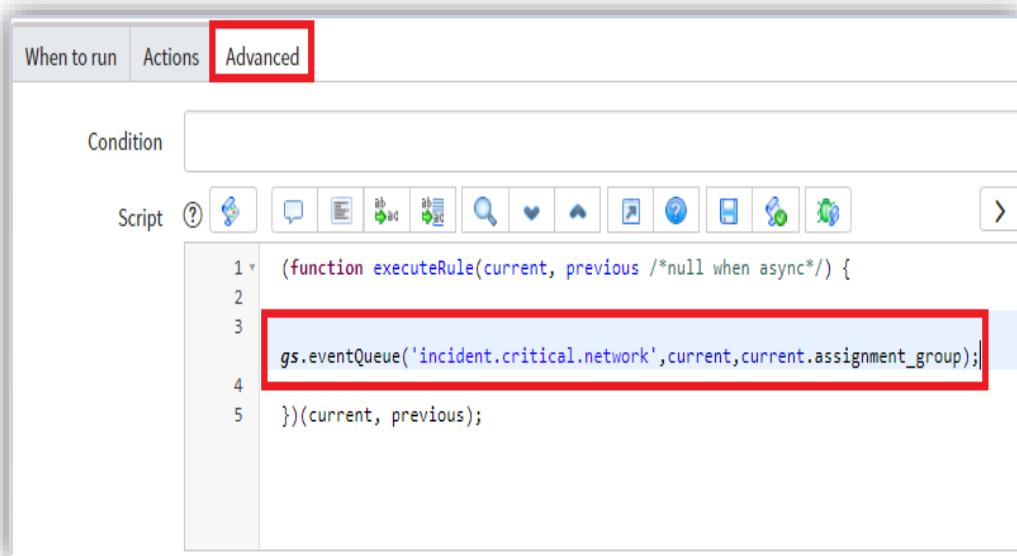
8. **When:** async
9. **Order:** 100
10. **Insert:** True

Filter Conditions

11. Priority is 1-Critical
12. Category is Network

Advanced

```
gs.eventQueue('incident.critical.network',current,current.assignment_group);
```



How to test Async business rules?

You can put some logs and see if those are printed in syslog table. You can also perform some actions in the script like updating any other record in another table and see if that record got updated or not. That way we can test Async business rule

Display Business Rules

Working with Display Business Rule

Display business rules are used to get data from server to client through callback function

Major Use Cases of Display BR:

1. The data is read from the database, display rules are executed, and the form is presented to the user.
2. The current object is available and represents the record retrieved from the database.
3. Any field changes are temporary since they are not yet submitted to the database.
4. To the client, the form values appear to be the values from the database; there is no indication that the values were modified from a display rule. This is a similar concept to calculated fields.
5. The primary objective of display rules is to use a shared scratchpad object, `g_scratchpad`, which is also sent to the client as part of the form.
6. This can be useful when you need to build client scripts that require server data that is not typically part of the record being displayed.
7. In most cases, this would require a client script making a call back to the server. If the data can be determined prior to the form being displayed, it is more efficient to provide the data to the client on the initial load.
8. The form scratchpad object is an empty object by default, and used only to store name:value pairs of data.
9. Storing field value for client.
10. Displaying message on the current form.



`g_scratchpad` variable is main object for display business rules

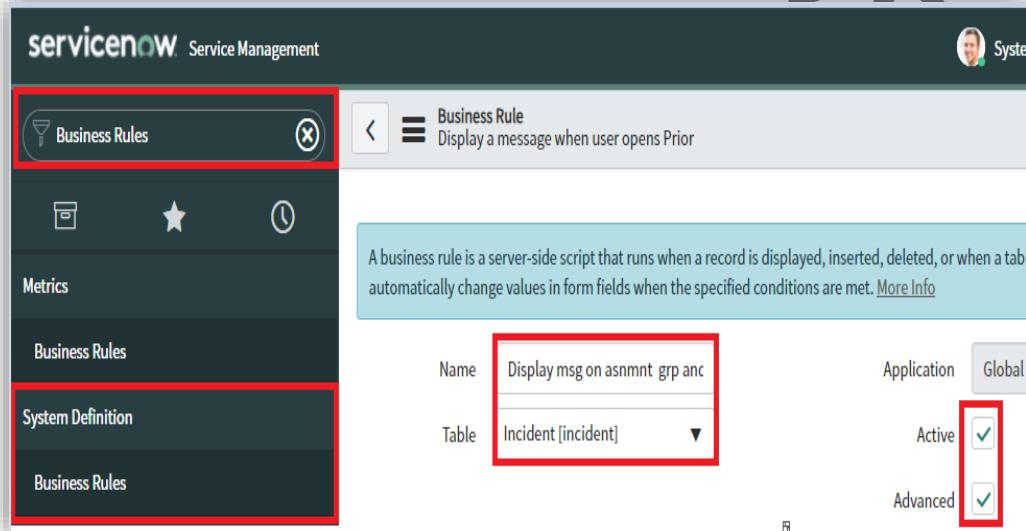
Display BR without Scripting:

Exercise: 15

Display an info message when user opens any incident record where the assignment group and assigned to field are empty.

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**



3. Fill the business rule form
4. **Name:** Display info message on assignment group and assigned to are empty
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

When to run Actions Advanced

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions run.

When display
Order 100

Filter Conditions Add Filter Condition Add "OR" Clause

All of these conditions must be met

Assignment group is empty AND OR X
Assigned to is empty AND OR X

```
graph TD; A[Assignment group is empty] -- AND --> B[Assigned to is empty]
```

When to run

8. When: display

9. Order: 100

Filter Conditions

10. Assignment Group is empty

11. Assigned to is empty

When to run Actions Advanced

Add message

Message

B I U Calibri 12pt

A A A A

Assignment group and Assigned to both are empty

```
graph TD; A[Assignment group and Assigned to both are empty]
```

Action

Add message: True

Message: Assignment group and Assigned to both are empty

12. Click on submit

Exercise: 16

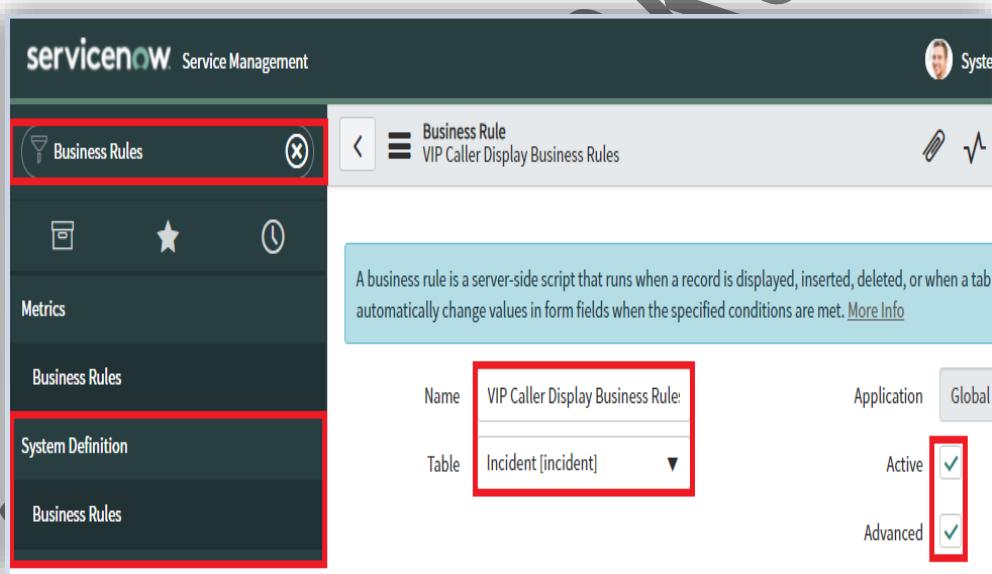
Display alert message that current caller is VIP or not

Create New Business Rule

Step1: Need to create new business rule

Procedure

1. Navigate to **System Definition→Business Rules**
2. Click on **New**



3. Fill the business rule form
4. **Name:** VIP Caller Display Business Rules
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

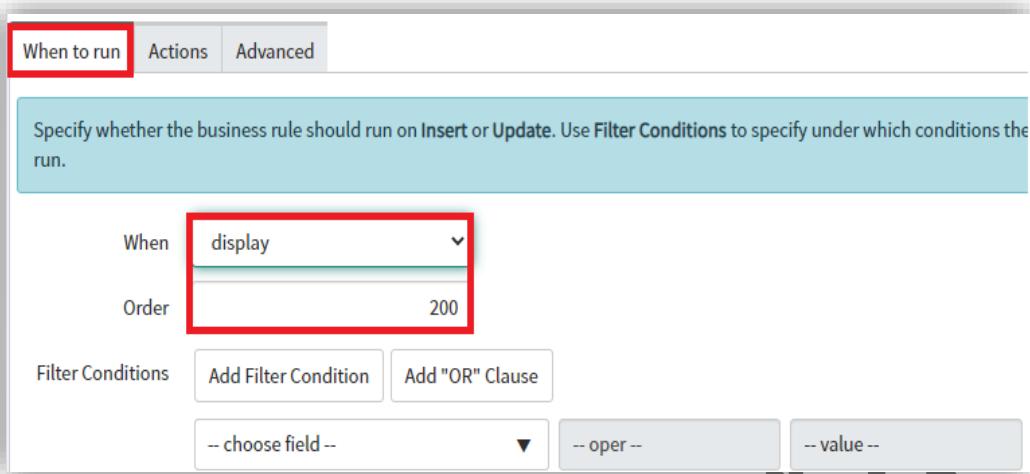
When to run Actions Advanced

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the rule runs.

When: display Order: 200

Filter Conditions Add Filter Condition Add "OR" Clause

-- choose field -- -- oper -- -- value --



When to run

8. When: display

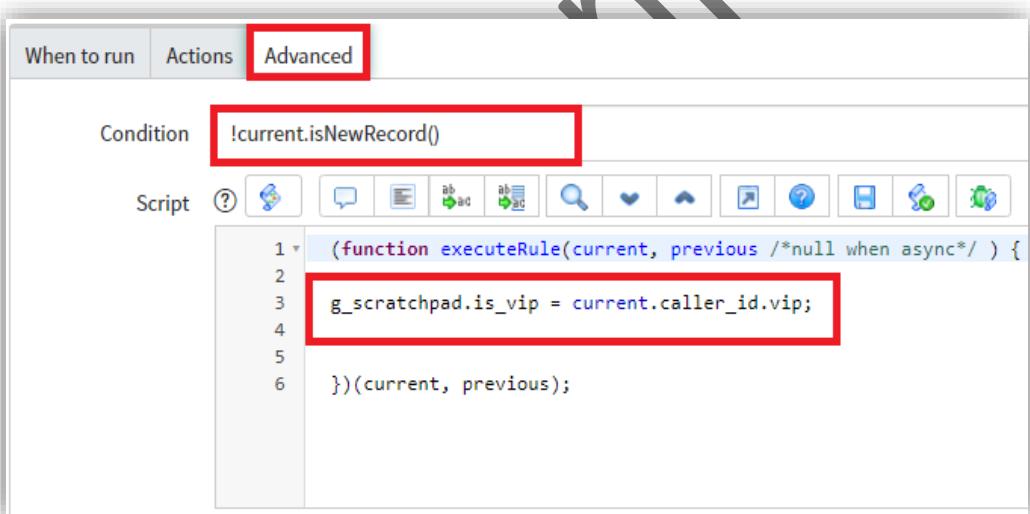
9. Order: 200

When to run Actions Advanced

Condition: !current.isNewRecord()

Script:

```
(function executeRule(current, previous /*null when async*/ ) {  
    g_scratchpad.is_vip = current.caller_id.vip;  
})(current, previous);
```



Advanced

10. Condition: !current.isNewRecord()

11. g_scratchpad.is_vip = current.caller_id.vip;

Create New Client Script

Step 2: Need to create new client script

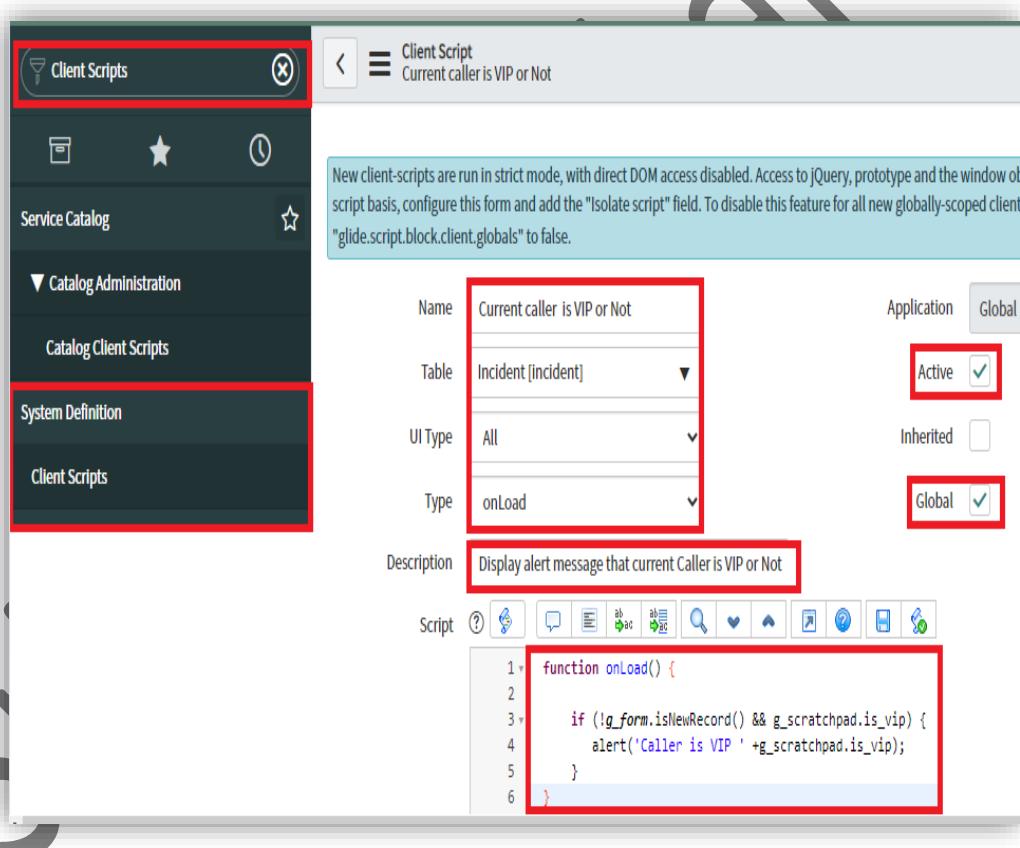
Procedure

1. Navigate to **System Definition** → **Client Scripts**
2. Click on **New**
3. Full fill client script form
4. **Name:** Current caller is VIP or Not
5. **Table:** Incident

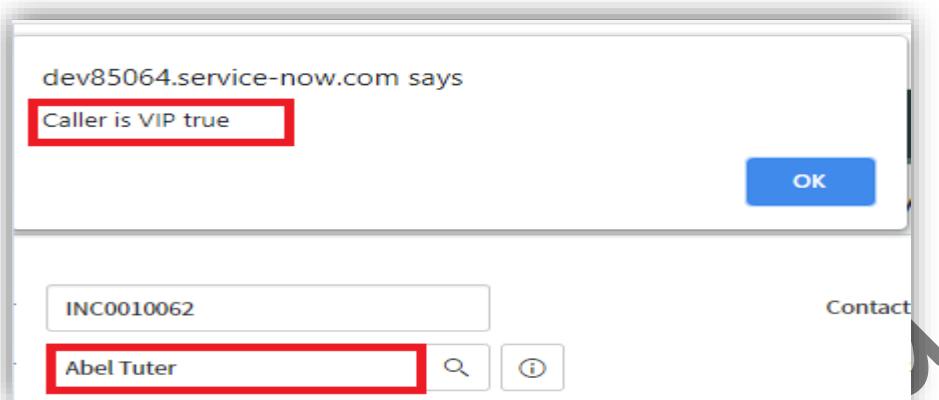
6. **UI Type:** All
7. **Type:** OnLoad
8. **Active:** True
9. **Global:** True
10. **Description:** Display alert message that current Caller is VIP or Not
11. **Script**

```
if (!g_form.isNewRecord() && g_scratchpad.is_vip) {
    alert('Caller is VIP ' + g_scratchpad.is_vip);
}
```

12. Click on **Submit**



13. Display Business Rule Result



Exercise: 17

Display current incident details like. Mail id, Priority and Category

Create New Business Rule

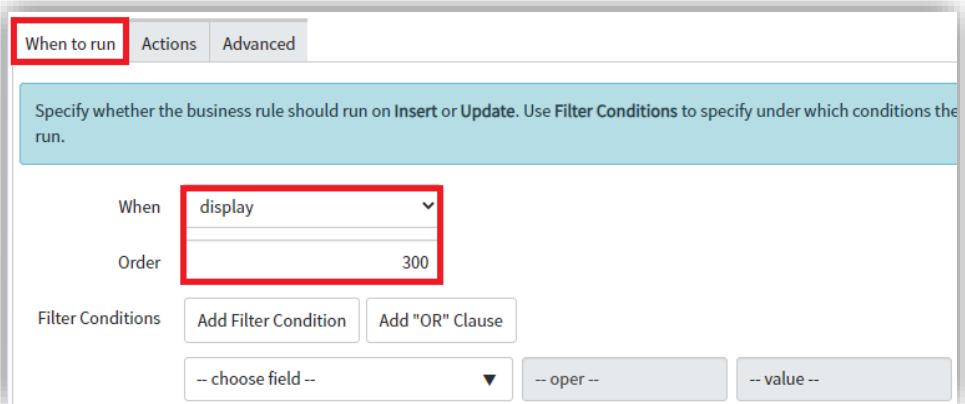
Step1: Need to create new business rule

Procedure

1. Navigate to System Definition → Business Rules
2. Click on New

A screenshot of the ServiceNow Business Rule creation form. The left sidebar shows "Business Rules" selected. The main panel title is "Business Rule" with the subtitle "Display current Incident details". A description box explains what a business rule is. The form fields are: Name (red box), Table (Incident [incident]), Application (Global), Active (checked), and Advanced (checked).

3. Fill the business rule form
4. **Name:** Display current Incident details
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True



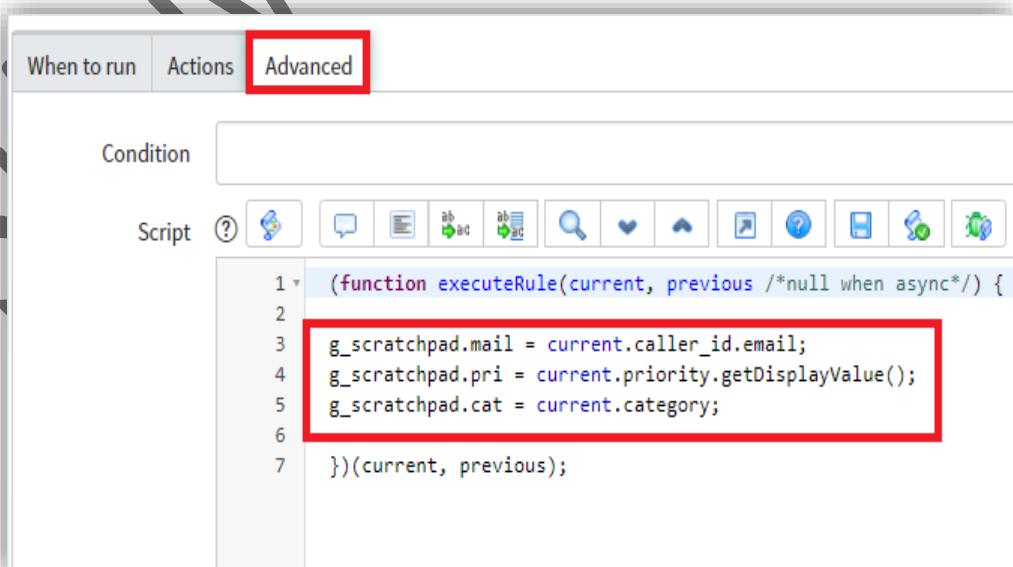
When to run

8. When: display
9. Order: 300

Advanced

10. Script

```
g_scratchpad.mail = current.caller_id.email;  
g_scratchpad.pri = current.priority.getDisplayValue();  
g_scratchpad.cat = current.category;
```



11. Click on Submit

Create New Client Script

Step 2: Need to create new client script

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Click on **New**
3. Full fill client script form
4. **Name:** Display incident details
5. **Table:** Incident
6. **UI Type:** All
7. **Type:** OnLoad
8. **Active:** True
9. **Global:** True
10. **Description:** Display incident details
11. **Script**

```
function onLoad() {  
  
    alert ('Current Caller is ' + g_scratchpad.mail);  
    alert ('Current Priority is ' + g_scratchpad.pri);  
    alert ('Current Category is ' + g_scratchpad.cat);  
}
```

The screenshot shows the ServiceNow interface for managing client scripts. The left sidebar has a red box around the 'Client Scripts' item under 'System Definition'. The main area shows a client script named 'Display incident details' for the 'Incident [incident]' table, set to run on 'onLoad'. The script code is highlighted with a red box and contains three alerts showing the current caller, priority, and category.

Exercise: 18

Display count of associated incidents for particular problem

Create New Business Rule

Step1: Need to create new business rule

Procedure

1. Navigate to System Definition → Business Rules
2. Click on New

The screenshot shows the ServiceNow Service Management interface. On the left, a sidebar menu is open with several items: Metrics, Business Rules, System Definition, and Business Rules. The 'Business Rules' item under the main 'Business Rules' heading is highlighted with a red box. The main content area is titled 'Business Rule' with the sub-title 'Display count of incident for Problem'. A descriptive text box explains what a business rule is. Below it, the rule details are shown: Name is 'Display count of incident for Problem', Table is 'Problem [problem]', Application is 'Global', Active is checked, and Advanced is checked. The entire 'When to run' section is also highlighted with a red box.

3. Fill the business rule form
4. **Name:** Display count of incident for Problem
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

This screenshot shows the 'When to run' configuration section of the business rule. It has three tabs: 'When to run' (which is selected and highlighted with a red box), 'Actions', and 'Advanced'. The main area asks 'Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the bus...'. Below this, there are two fields: 'When' (set to 'display') and 'Order' (set to '400'). There are also buttons for 'Add Filter Condition' and 'Add "OR" Clause'. At the bottom, there are dropdowns for 'choose field', 'oper', and 'value'.

When to run

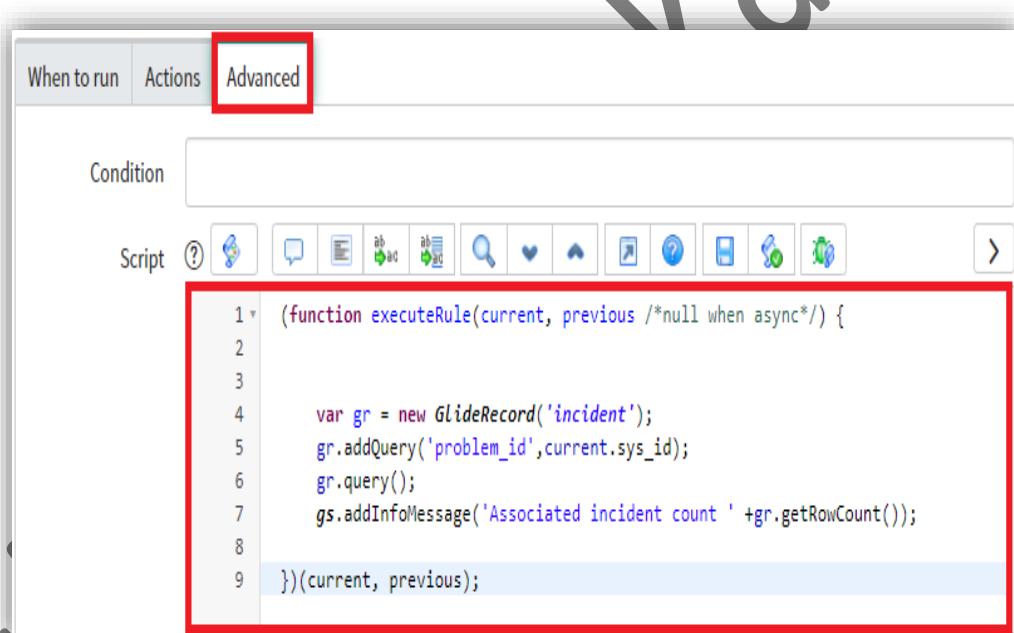
8. **When:** display
9. **Order:** 400

Advanced

10. Script

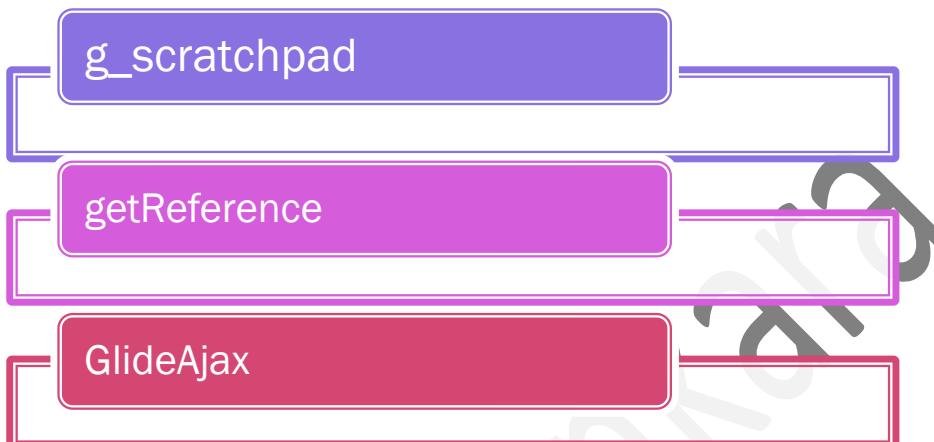
```
var gr = new GlideRecord('incident');
gr.addQuery('problem_id',current.sys_id);
gr.query();
gs.addInfoMessage ('Associated incident count ' +
gr.getRowCount());
```

11. Click on Submit



How many ways you will get data from Server to Client

1. We are going to getting the data from **Server to Client** very frequently
2. Basically we have **3 ways** of getting data from server to client



Working with **g_scratchpad** variable

Exercise: 19

Display current record category

Create New Business Rule

Step1: Need to create new display business rule

Procedure

1. Navigate to **System Definition → Business Rules**
2. Click on **New**

The screenshot shows the 'Business Rules' application interface. On the left, there's a sidebar with 'Metrics', 'Business Rules', 'System Definition' (which is highlighted with a red box), and 'Business Rules'. The main area has a header 'Business Rule' and 'New record'. A descriptive text box explains what a business rule is. The configuration form includes fields for 'Name' (set to 'g_scratchpad variable example'), 'Table' (set to 'Incident [incident]'), 'Application' (set to 'Global'), 'Active' (checkbox checked), and 'Advanced' (checkbox checked). A large watermark 'Sunkar' is diagonally across the page.

3. Fill the business rule form
4. **Name:** g_scratchpad variable example
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

The screenshot shows the 'When to run' tab of the business rule configuration. It has tabs for 'When to run', 'Actions', and 'Advanced'. The 'When to run' tab contains a note about specifying when the rule runs on 'Insert' or 'Update'. It has fields for 'When' (set to 'display') and 'Order' (set to '500'). Below these are 'Filter Conditions' buttons for 'Add Filter Condition' and 'Add "OR" Clause'. A dropdown menu for 'Priority' is open, showing options: '1 - Critical', '2 - High', '3 - Moderate', '4 - Low', and '5 - Info'. The '1 - Critical' option is highlighted with a red box. A large watermark 'Sunkar' is diagonally across the page.

When to run

8. When: display
9. Order: 500

Filter Conditions

Priority is one of 1-Critical or 2-High

Advanced

The screenshot shows a software interface for creating client scripts. At the top, there are three tabs: 'When to run', 'Actions', and 'Advanced'. The 'Advanced' tab is currently selected and highlighted with a red box. Below the tabs is a section labeled 'Condition' with a dropdown menu. Underneath is a toolbar with various icons, followed by a script editor area. The script editor contains the following code:

```
(function executeRule(current, previous /*null when async*/) {  
    g_scratchpad.cat = current.category;  
})(current, previous);
```

A large red box surrounds the entire script editor area.

Create New Client Script

Step 2: Need to create new client script

Procedure

1. Navigate to System Definition→Client Scripts
2. Click on New
3. Full fill client script form
4. Name: Display current record category
5. Table: Incident
6. UI Type: All
7. Type: OnLoad
8. Active: True
9. Global: True
10. Description: Display current record category through scratchpad variable
11. Script

```
function onLoad () {  
    alert ('Current record category is ' +  
        g_scratchpad.cat);  
}
```

The screenshot shows the ServiceNow Catalog Client Scripts editor. The left sidebar has a red box around the 'Client Scripts' item under 'System Definition'. The main area shows a client script named 'Display current record category' for the 'Incident [incident]' table. The 'Type' is set to 'onLoad'. The 'Active' checkbox is checked, and the 'Global' checkbox is checked. The script code is:

```
function onLoad() {  
    alert('Current record category is ' + g_scratchpad.cat);  
}
```

12. Click on Submit

Working with getReference

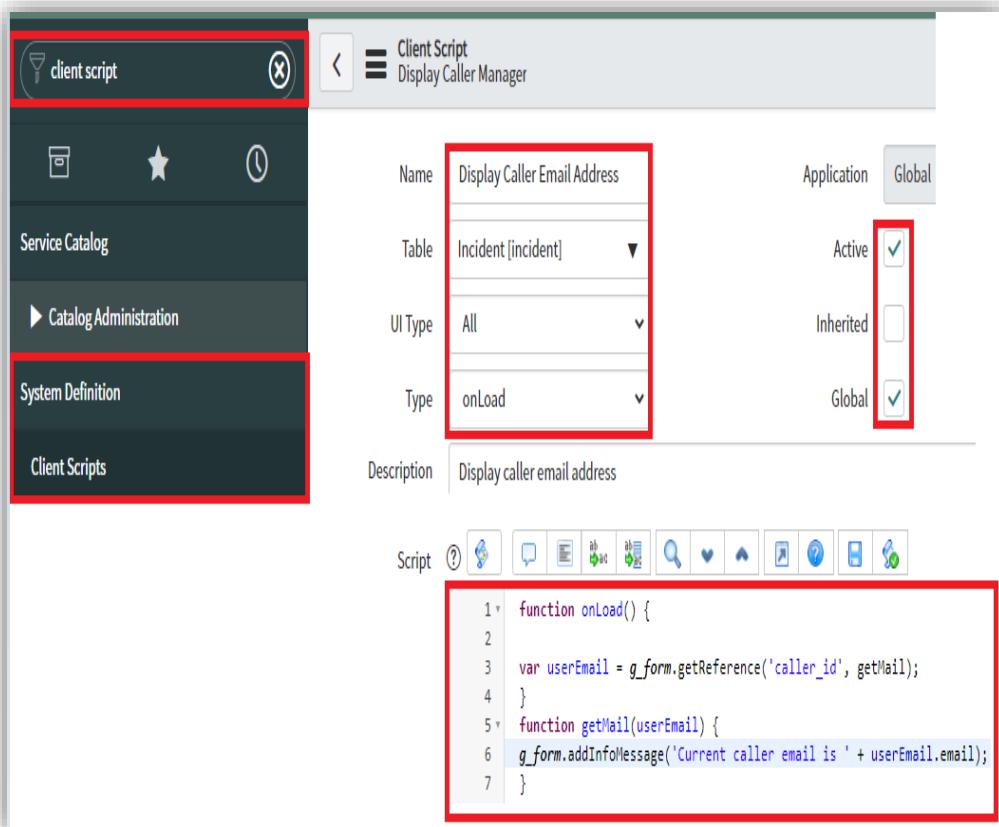
Exercise: 20

Display current record caller mail address

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Click on **New**
3. Full fill client script form
4. **Name:** Display current caller email address
5. **Table:** Incident
6. **UI Type:** All
7. **Type:** OnLoad
8. **Active:** True
9. **Global:** True
10. **Description:** Display current caller email address
11. **Script**

```
function onLoad() {  
    var userEmail = g_form.getReference('caller_id', getMail);  
}  
function getMail(userEmail) {  
    g_form.addInfoMessage('Current caller email is ' +  
    userEmail.email);  
}
```



Script Include

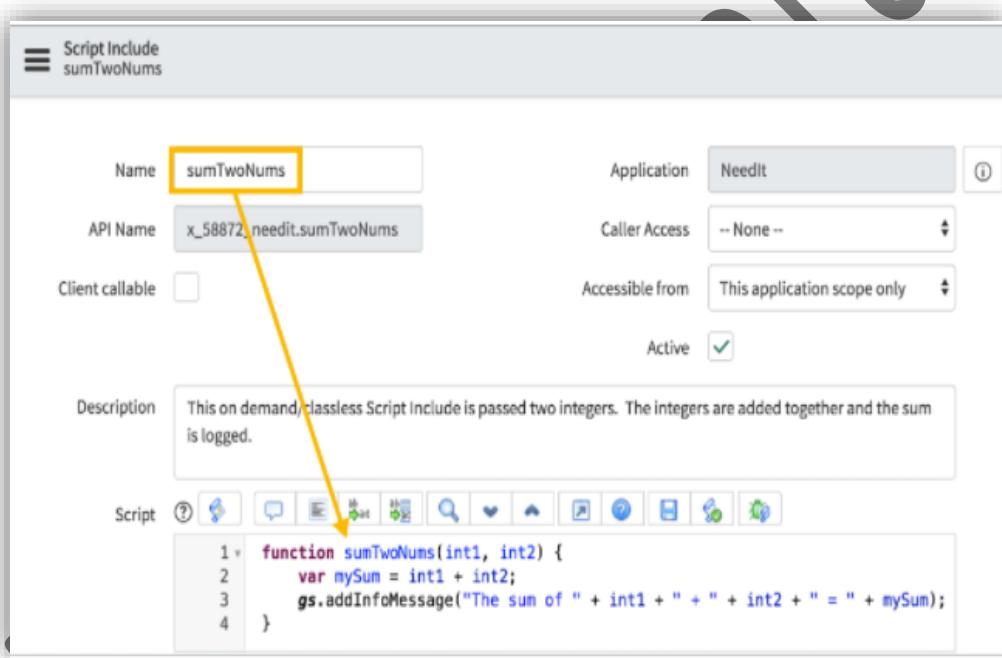
What is Script Include?

Script Includes are reusable server-side script logic that define a function or class. Script Includes execute their script logic only when explicitly called by other scripts. There are different types of Script Includes

1. On demand/classless
2. Extend an existing class
3. Define a new class

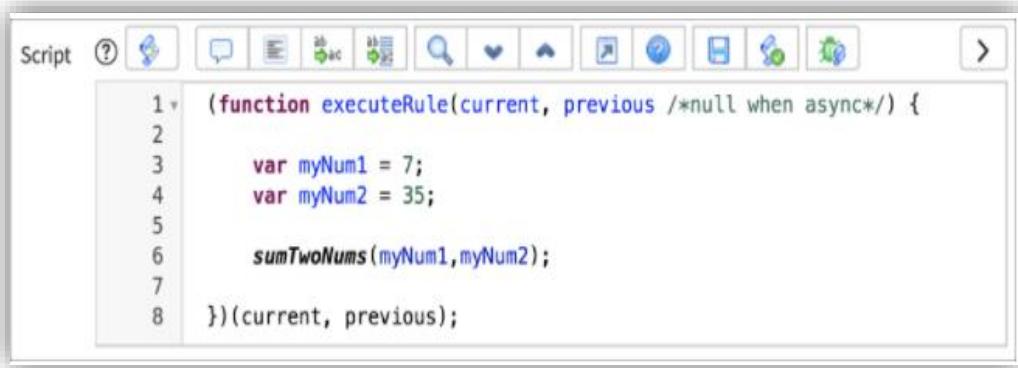
On Demand Script Include

- ✓ A Script Include that defines a single function is known as an on demand, or classless, Script Include.
- ✓ The function is callable from other server-side scripts. On demand Script Includes can never be used client-side even if the Client callable option is selected.
- ✓ A script template is automatically inserted into the Script field.
- ✓ The template does not apply to on demand Script Includes. Delete the template and replace it with your function definition.
- ✓ The Script Include function is defined using standard JavaScript syntax. The example shows an on demand Script Include:



The Script Include name must exactly match the name of the function. In the example, both the Script Include and the function are named **sumTwoNums**.

The on demand Script Include is usable by any other server-side script allowed by the Accessible from option. The example Business Rule script uses the **sumTwoNums** on demand Script Include:



The screenshot shows a 'Script' editor window with a toolbar at the top containing various icons. The script code is as follows:

```
1 (function executeRule(current, previous /*null when async*/) {  
2     var myNum1 = 7;  
3     var myNum2 = 35;  
4  
5     sumTwoNums(myNum1,myNum2);  
6  
7 })(current, previous);
```

Although the **sumTwoNums** function is not defined in the Business Rule script, the function exists because it is defined in the Script Include.

(i) The sum of 7 + 35 = 42 X

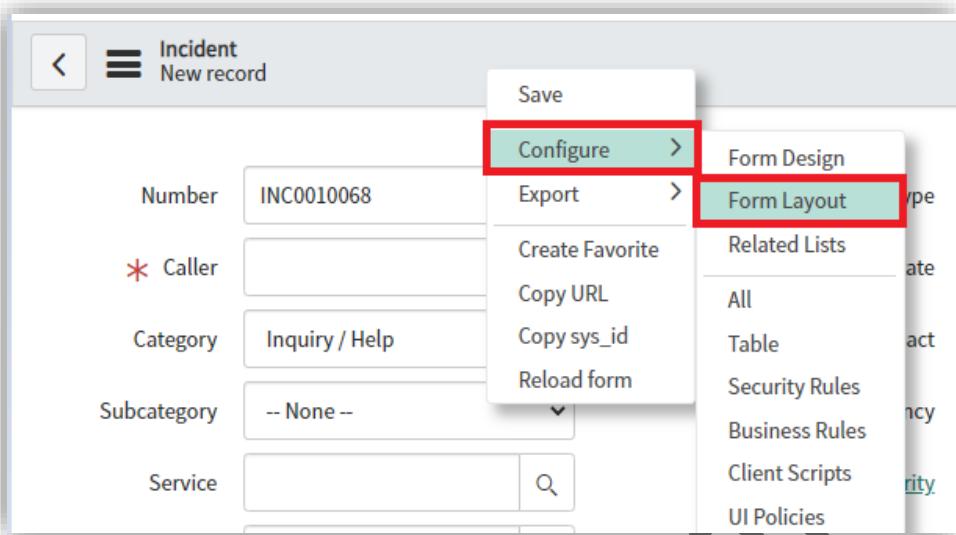
On demand Script Includes are typically used when script logic needs to be reused. Examples include standardizing date formats, enabling/disabling logging, and validating email addresses.

Working with On Demand Script Include

We should validate email address. The Script Include can be used by any application

Procedure

1. Add a string field (**Requested for email**) to the Incident table to store an **email address**.
2. Use the **Configure → Form Layout**



3. Create new field
4. Click on Add and Save

A screenshot of the 'Create new field' dialog. It has three input fields: 'Name' (Requested For Email), 'Type' (String), and 'Field length' (Small (40)). The 'Add' button at the bottom left is highlighted with a red box.

5. Field make Mandatory

A screenshot of the assignment group form. It includes fields for 'Assignment group', 'Assigned to', 'Start Date', 'End Date', and 'Requested For Email'. The 'Requested For Email' field is marked with a red asterisk (*) and highlighted with a red box, indicating it is a mandatory field.

Create New On Demand Script Include

Need to create new script include

Procedure

1. Navigate to **System UI→Script Include**
2. Click on **New**
3. **Name:** validateEmailAddress
4. **API Name:** global.validateEmailAddress
5. **Accessible from:** This application scope only
6. **Active:** True
7. **Script**

```
function validateEmailAddress(emailStr){  
    // Use JavaScript coercion to guarantee emailStr is a string  
    emailStr = emailStr + ";  
    // Compare emailStr against the allowed syntax as specified in the  
    // regular expression  
    // If emailStr has allowed syntax, return true, else return false  
  
    if(emailStr.match(/^(([^\<>()\[\]\.,;:\s@"]+(\.[^\<>()\[\]\.,;:\s@"]+)|  
        ("."+))@((\[[0-9]{1,3}\].[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3})|(([a-zA-Z]-0-  
        9]+\.)+[a-zA-Z]{2,}))$/)){  
        return true;  
    }  
    else {  
        return false;  
    }  
}
```

Script include

validateEmailAddress

Name: validateEmailAddress

Application: Global

Accessible from: This application scope only

Active:

Script:

```
function validateEmailAddress(emailStr) {  
    // Use JavaScript coercion to guarantee emailStr is a string  
    emailStr = emailStr + '';  
    // Compare emailStr against the allowed syntax as specified in the regular expression  
    // If emailStr has allowed syntax, return true, else return false  
    if (emailStr.match(/^\w+((\.\w+){0,1})?@[a-zA-Z0-9]+\.(a-zA-Z0-9)+\.(a-zA-Z0-9)+$/)) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

Use the Script Include in a Business Rule

Need to create new business rule

Procedure

- 1. Navigate to System Definition → Business Rules
- 2. Click on New

Business Rules

Email Address Syntax Validate

Name: Email Address Syntax Validate

Table: Incident [incident]

Application: Global

Active:

Advanced:

3. Fill the business rule form
4. **Name:** Email Address Syntax Validation
5. **Table:** Incident
6. **Active:** true
7. **Advanced:** True

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which circumstances the rule should run.

When	before
Order	100
Insert	<input checked="" type="checkbox"/>
Update	<input checked="" type="checkbox"/>
Delete	<input type="checkbox"/>
Query	<input type="checkbox"/>

When to run

8. **When:** before
9. **Order:** 100
10. **Insert:** true
11. **Update:** true

Advanced

12. Script

```
var isEmail = validateEmailAddress(current.u_requested_for_email); //  
If isEmail is false (email address syntax is not valid) do not save // the  
record. Write an error message to the screen.  
if(isEmail == false){  
    gs.addErrorMessage(current.u_requested_for_email + " is not a valid  
    email address. You must provide a valid email address.");  
    current.setAbortAction(true);  
}
```

When to run Actions **Advanced**

Condition

Script >

```
1 * (function executeRule(current, previous /*null when async*/){  
2  
3     var isEmail = validateEmailAddress(current.u_requested_for_email);  
4     if(isEmail == false){  
5         gs.addErrorMessage(current.u_requested_for_email + " is not a valid  
email address. You must provide a valid email address.");  
6         current.setAbortAction(true);  
7     }  
8 }))(current, previous);
```

Glide Ajax

Glide Ajax Overview

The Glide Ajax class enables a client script to call server-side code in a script include.

To use Glide Ajax in a client script, follow these general steps.

1. Create a Glide Ajax instance by calling the Glide Ajax constructor. As the argument to the constructor, specify the name of the script include class that contains the method you want to call.
2. Call the addParam method with the **sysparm_name** parameter and the name of the script-include method you want to call.
3. (Optional) Call the addParam method one or more times to provide the script-include code with other parameters it needs.
4. Execute the server-side code by calling **getXML()**.

Note: `getXML()` is the preferred method for executing the code, because it is asynchronous and does not hold up the execution of other client code.

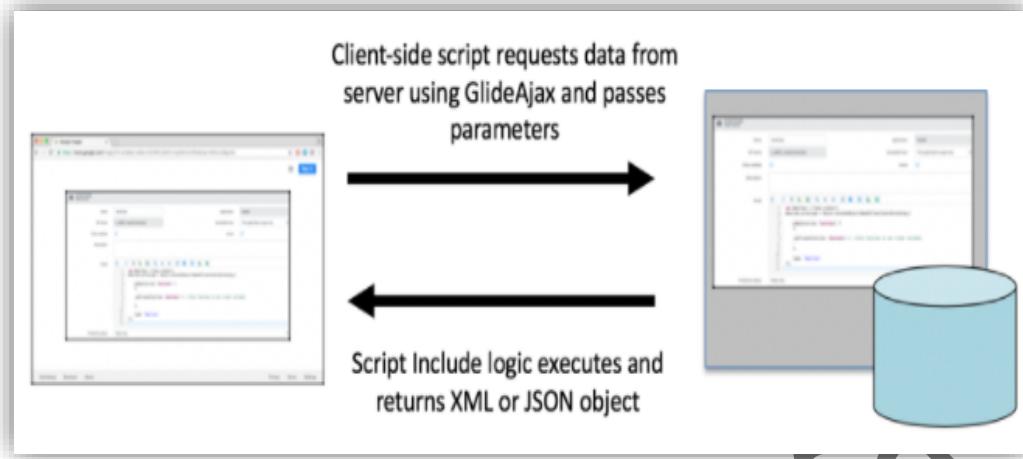
Another method, **getXMLWait()**, is also available but is not recommended. Using **getXMLWait ()** ensures the order of execution, but can cause the application to seem unresponsive, significantly degrading the user experience of any application that uses it. **getXMLWait ()** is not available to scoped applications.

Glide Ajax Syntax Example

```
var ga = new GlideAjax('HelloWorld'); // HelloWorld is the script include class  
ga.addParam('sysparm_name','helloWorld'); // helloWorld is the script include  
method  
ga.addParam('sysparm_user_name','Bob'); // Set parameter  
sysparm_user_name to 'Bob'  
ga.getXML(HelloWorldParse); /* Call HelloWorld.helloWorld() with the  
parameter sysparm_user_name set to 'Bob'  
and use the callback function HelloWorldParse() to return the result when  
ready */  
  
// the callback function for returning the result from the server-side code  
function HelloWorldParse(response) {  
    var answer =  
    response.responseXML.documentElement.getAttribute("answer");  
    alert(answer);  
}
```

Extending GlideAjax

The **GlideAjax** class is used by client-side scripts to send data to and receive data from the ServiceNow server. The client-side script passes parameters to the **Script Include**. The Script Include returns data as **XML or a JSON object**. The client-side script parses data from the response and can use the data in subsequent script logic.



The incident form has a reference field called *Requested for*. The *Requested for* field references a record on the *User [sys_user]* table. The *User* table has a column called *Email*, which stores a User's email address.

Recall that client-side scripts have access only to data from the fields on a form. Although the *incident* form has a field that references the *User* table, it does not have access to the *Requested for*'s email address which is stored in the database. This example will extend the *GlideAjax* class. The new class will be passed a *sys_id* for the *User* table and will retrieve and pass back the user's email address.

GlideAjax is used by *g_form* methods like **getReference()**.

The **getReference()** method is passed a *sys_id* and returns that record as a **GlideRecord**. The new class created in this demonstration is passed a *sys_id* and returns only an email address and not an entire record.

Although the performance difference between returning an entire record and a single value may seem negligible, performance is based on all the calls occurring on a form and not a single call. If multiple scripts save time, the cumulative effect can be noticeable

Working with Script Include, Glide Ajax and Client Script

The Script Include must be client callable because it will be used by client-side scripts.

The **AbstractAjaxProcessor** class is part of the *Global Scope*.

The **GetEmailAddress** Script Include is in the *incident*. To extend the *AbstractAjaxProcessor*, the class must be prepended by the

scope: `global.AbstractAjaxProcessor`. The new class defines a method called `getEmail`.

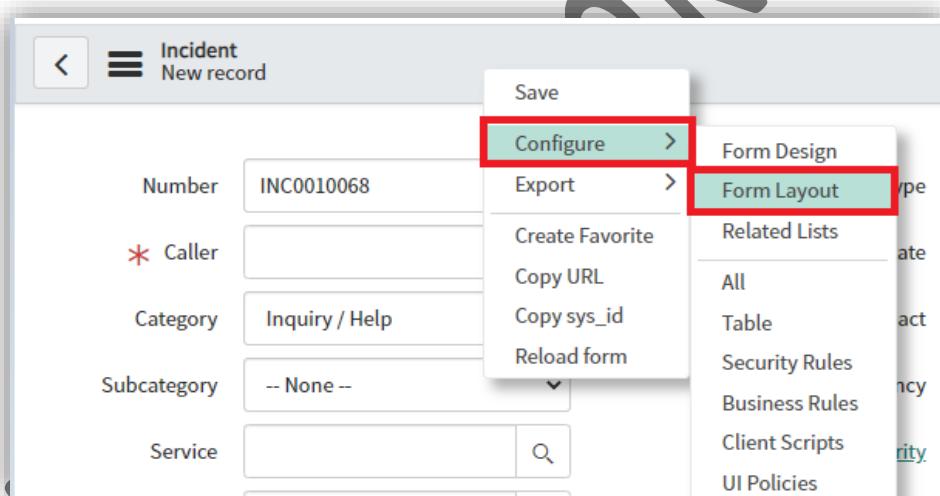
Excercise:21

We should write a Script Include to create a client-callable Script Include by extending `GlideAjax`. The new Script Include retrieves the **Requested for's email** address from the User table.

Let's start create new Script Include (**Extend an existing class**)

Procedure

1. Add two more fields (**Requested for**) and (**Requested for email**) to the *Incident* table to get an **email address**.
2. Use the **Configure** → **Form Layout**



3. Create new field(Reference)
4. Click on Add and Save

A screenshot of the 'Create new field' dialog box. It shows a 'Save' button at the top left. The form has three fields: 'Name' (set to 'Requested For'), 'Type' (set to 'Reference'), and 'Table to reference' (set to 'User [sys_user]'). The 'Type' and 'Table to reference' fields are highlighted with red boxes. At the bottom left is an 'Add' button.

Name	Requested For
Type	Reference
Table to reference	User [sys_user]

5. Create one more string field (Requested for email)

The screenshot shows a 'Create new field' dialog box. At the top right is a red 'Save' button. Below it is a table with three rows: 'Name' (containing 'Requested For Email'), 'Type' (containing 'String'), and 'Field length' (containing 'Small (40)'). All three rows have a red border around them. At the bottom left is a red 'Add' button.

Working with Script Include

We should create new Script Include

Procedure

1. Navigate to System UI → Script Include
2. Click on New
3. Name: getEmailAddress
4. API Name: Auto filled
5. Accessible from: This application scope only
6. Active: True
7. Script
8. Description: Script Include to return an email address. The calling client-side script passes a **sys_id** for a User table record.
9. Client callable: True
10. Copy entire code template and paste into script field
11. Script

```
var getEmailAddress = Class.create();
// Extend the global.AbstractAjaxProcessor class
getEmailAddress.prototype =
Object.extendsObject(global.AbstractAjaxProcessor,{
    // Define the getEmail function.
    // Create a GlideRecord for the User table.
    // Use the sysparm_userID passed from the client side to
    // retrieve a record from the User table.
    // Return the email address for the requested record
    getEmail: function() {
        var userRecord = new GlideRecord("sys_user");
        userRecord.get(this.getParameter('sysparm_userID'));
        return userRecord.email + "";
    },
});
```

```
type: 'GetEmailAddress'  
});
```

The screenshot shows a client script editor window with a toolbar at the top containing various icons. The main area displays the following JavaScript code:

```
1 var GetEmailAddress = Class.create();
2 // Extend the global.AbstractAjaxProcessor class
3 GetEmailAddress.prototype = Object.extendsObject(global.AbstractAjaxProcessor,{
4     // Define the getEmail function.
5     // Create a GlideRecord for the User table.
6     // Use the sysparm_userID passed from the client side to retrieve a record
7     // from the User table.
8     getEmail: function() {
9         var userRecord = new GlideRecord("sys_user");
10        userRecord.get(this.getParameter('sysparm_userID'));
11        return userRecord.email + '';
12    },
13    type: 'GetEmailAddress'
14 });
15
```

A specific line of code, `userRecord.get(this.getParameter('sysparm_userID'));`, is highlighted with a light blue background.

Working with Client Script and GlideAjax

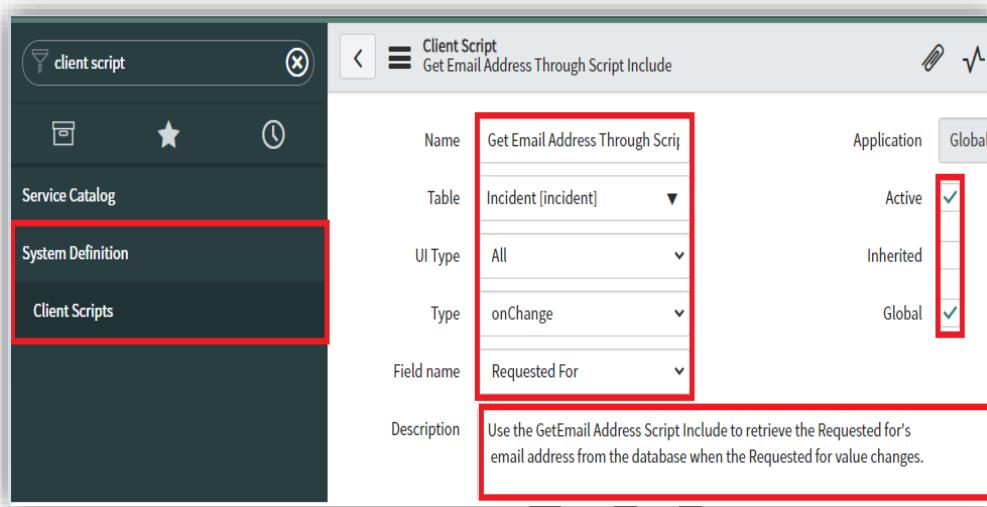
We should create new client script

Procedure

1. Navigate to **System Definition→Client Scripts**
2. Click on **New**
3. Full fill client script form
4. **Name:** Get Email Address Through Script Include
5. **Table:** Incident
6. **UI Type:** All
7. **Type:** OnChange
8. **Active:** True

9. Global: True

- 10. Description:** Use the GetEmail Address Script Include to retrieve the Requested for email address from the database when the Requested for value changes.



11. Script

```
function onChange (control, oldValue, newValue, isLoading, isTemplate) {  
    if (isLoading || newValue === "") {  
        return;  
    }  
  
    var getEmailAddr = new GlideAjax('GetEmailAddress');  
    // Specify the getEmail method  
    getEmailAddr.addParam('sysparm_name','getEmail');  
    // Pass the Requested for sys_id  
    getEmailAddr.addParam('sysparm_userID',  
        g_form.getValue('u_requested_for'));  
    // Send the request to the server  
    getEmailAddr.getXML(populateEmailField);  
    // When the response is back from the server  
    function populateEmailField(response){  
        // Extract the email address from the response, clear any value  
        // from the email field,  
        // set new value in the email field  
        var emailFromScriptInclude =  
            response.responseXML.documentElement.getAttribute("answer");  
        g_form.clearValue('u_requested_for_email');  
  
        g_form.setValue('u_requested_for_email',emailFromScriptInclude);  
    }  
}
```

```
        g_form.setDisabled('u_requested_for_email', true);
    }
}
```

12. Script

```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading || newValue === '') {
        return;
    }

    var getEmailAddr = new GlideAjax('GetEmailAddress');
    // Specify the getEmail method
    getEmailAddr.addParam('sysparm_name','getEmail');
    // Pass the Requested for sys_id
    getEmailAddr.addParam('sysparm_userID', g_form.getValue('u_requested_for'));
    // Send the request to the server
    getEmailAddr.getXML(populateEmailField);

    // When the response is back from the server
    function populateEmailField(response){
        // Extract the email address from the response, clear any value from the
        email field,
        // set new value in the email field
        var emailFromScriptInclude =
        response.responseXML.documentElement.getAttribute("answer");
        g_form.clearValue('u_requested_for_email');
        g_form.setValue('u_requested_for_email',emailFromScriptInclude);
        g_form.setDisabled('u_requested_for_email', true);
    }
}
```

Test Script Include and Client Script

1. Switch to the main ServiceNow browser window and use the browser reload button to reload ServiceNow.
2. Open an existing incident record for editing (for this part of the exercise, do not create a new record).
3. Change the value in the Requested for field to **Fred Luddy**.
4. The Requested for email field value should change to **fred.luddy@example.com**. If not, debug using the debugging strategies explained earlier in this module. If you see a message about cross-scope privileges, it was caused by ServiceNow detecting allowed

- use of an out-of-scope file (`AbstractAjaxProcessor`). You will see the message only once.
5. Change the value in the Requested for field to **Beth Anglin**. The value in the Requested for email field should change again and display **Beth Anglin**.

The screenshot shows a search interface with the following fields and results:

- Priority:** 5 - Planning
- Assignment group:** [Search input]
- Assigned to:** [Search input]
- Requested For:** **Beth Anglin** (highlighted with a red box)
- * Requested For Email:** beth.anglin@example.com

► **QUESTION:** How does ServiceNow know which Script Include to call?
How does it know which method to use?

ANSWER: To identify the Script Include, pass the Script Include name when instantiating the `GlideAjax` object. The `addParam()` method selects the method to use from the Script Include based on the value passed as `sysparm_name`.

Core Important Concepts:

1. Server-side scripts execute on the ServiceNow server and have access to the database
2. Business Rules are triggered by database operations: query, update, insert, and delete

Server-side script APIs include:

1. `GlideRecord`
2. `GlideSystem`

- 3. Glide Session
- 4. Glide date
- 5. GlideDateTime
- 6. Glide Aggregation

Business Rule script logic is executed relative to when the database operation occurs

- 1. before
- 2. after
- 3. async
- 4. display

Debug Business Rules using:

- 1. JavaScript Debugger - debug script logic
- 2. Debug Business Rule (Details) - debug condition script
- 3. Application log - view log messages
- 4. Script Tracer - Determine which server-side scripts execute as part of a UI interaction

Use the Script Tracer to find information about:

- 5. State
- 6. Script
- 7. Transaction
- 8. Use the JavaScript Debugger to debug synchronous server-side scripts
- 9. See variable values
- 10. Set breakpoints
- 11. Set logpoints
- 12. Use the Console to evaluate expressions in the runtime environment
- 13. View call stack
- 14. See transaction information
- 15. Script Includes are reusable server-side logic

On demand/classless Script Includes

- 1. Cannot be called from the client-side
- 2. Contain a single function

Script Includes which extend a class

- 1. Inherit properties of extended class

2. Do not override inherited properties
3. Most commonly extended class is GlideAjax

Script Includes which create a new class (don't extend an existing class)

1. Many applications have a Utils Script Include
2. Initialize function automatically invoked
3. Developers must create all properties

crinivas Sunkara

Interview Questions on Business Rules

QUESTION:1

Which of these classes are part of the ServiceNow server-side API?

More than one response may be correct.

1. GlideSystem (gs)
2. GlideUser (g_user)
3. GlideDateTime
4. GlideDate
5. GlideForm (g_form)

ANSWER: Responses **1**, **3**, and **4** are correct. **GlideSystem**, **GlideDateTime**, and **GlideDate** are part of the ServiceNow server-side API. The server-side API also has a GlideUser class but the server-side GlideUser class does not use the g_user object. If you are not sure whether a class is part of the client-side or server-side API, check the [API Reference](#).

QUESTION:2

Which one of the following describes when before Business Rules execute their script logic?

1. Before a form loads
2. After a form loads but before control is given to the user
3. Before onChange Client Scripts
4. Before Business Rule Actions
5. Before records are written to the database

ANSWER: onBefore Business Rules execute their script logic before records are updated in the database.

QUESTION:3

What is the difference between an after Business Rule and an async Business Rule?

ANSWER: Both after and async Business Rules execute their script logic after records are written to the database. after Business Rules execute their logic immediately after a record is written to the database. async Business Rules create scheduled jobs that run soon after a record is written to the database.

QUESTION:4

Which of the following are NOT methods from the *GlideRecord* API? More than one response may be correct.

1. addQuery ()
2. addEncodedQuery()
3. addOrQuery()
4. addAndQuery()
5. query()

ANSWER: Responses **3** and **4** are correct. addOrQuery () and addAndQuery () are not methods from the GlideRecord API. If a script contains multiple

statements that use the addQuery () method the queries are ANDed. To explicitly AND or OR a condition in a query, use the methods from the GlideQueryCondition class.

QUESTION:5

Which of the following are NOT true about the current object? More than one response may be correct.

1. The current object is automatically instantiated.
2. The current object property values never change after a record is loaded from the database.
3. The current and previous objects are always identical.
4. The current and previous objects are sometimes identical.
5. The properties of the current object are the same for all Business Rules.

ANSWER: Responses **2, 3, and 5** are correct. Although the current object's property values do not have to change, they can. The current object's property values are sometimes identical to the previous object's properties. For example, the current and previous objects are identical immediately after a record is loaded from the database. The properties of the current object for Business Rules are dependent on table the Business Rule is for.

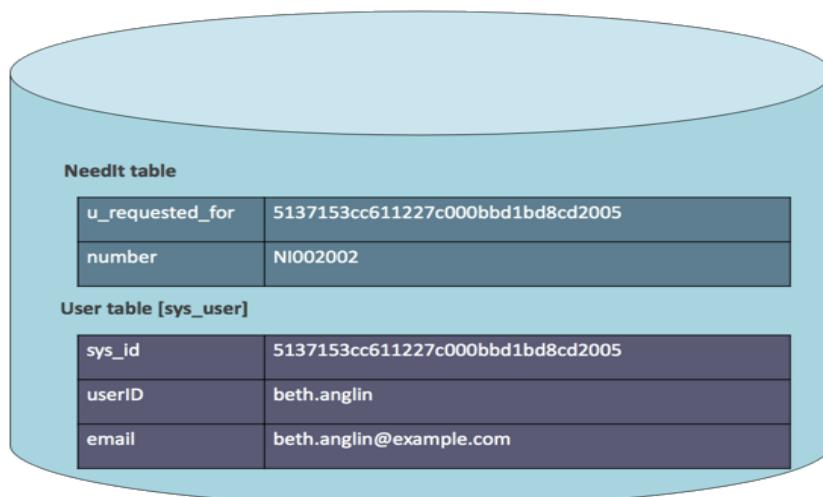
QUESTION:6 What value does a Business Rule Condition field return if the field is empty?

1. true
2. false
3. Nothing
4. ""
5. NULL

ANSWER: Response 1 is correct. If there is no value in the Condition field, the field returns true. Business Rule scripts only execute their script logic if the Condition field returns true.

QUESTION:7

Examine the database tables and fields.



Based on the database, which one of the following is valid dot-walking syntax?

1. u_requested_for.userID
2. current.u_requested_for.userID
3. number.userID
4. current.number.userID
5. previous.email.u_requested_for

ANSWER: Response 2 is correct. Dot-walking allows direct scripting access to fields and field values on related records.

The dot-walking syntax is:

object.related_object.field_name

QUESTION:8

Which of the following are true about Script Includes? More than one response may be correct.

1. Script Includes are reusable server-side script logic
2. Script includes can extend an existing class
3. Script includes can define a new class or function
4. Script includes can be client callable

5. Script includes execute their script logic only when explicitly called

ANSWER: Responses **1, 2, 3, 4, and 5** are correct. All of the statements are true for Script Includes

Srinivas Sunkara

Client Side Components

- Client Scripts
- UI Policy
- UI Scripts
- Catalog Client Script
- Catalog UI Policy
- Widget client script
- UI Pages - Client script

Server Side Component

- Business rule
- Script include
- Script action
- Schedule job
- Fix Scripts
- Workflow run script
- Email Scripts
- Installation Exits
- Widget server side script
- ACL Script
- Transform map scripts
- Field map scripts
- UI Actions
- UI Pages - Processing script
- UI Macros

Prinivas Sunkara

Lesson-13

Flow Designer

Agenda

1. Flow Designer Overview
2. Workflow Engine VS. Flow Designer
3. Architecture of Flow Designer
4. Sequence of Flow Process
5. Process Flow Execution Diagram
6. Different States In flow designer
7. Flow designer Features and Advantages
8. What is Flow?
9. Triggers and Types of Triggers
10. Action Flow and Action flow Order
11. Practice with more examples
12. Working with Integration HUB

Flow Designer Overview

Flow Designer is an application within ServiceNow platform this is new in the **Kingston release**.

Main aim of this application the business owners use natural language to automate business process, **approvals, notifications, record operations, and task executions**. Enables you lower risk for automation and accelerate development by ServiceNow developers.

Flow Designer provide us rich capabilities for automating processes to reduce repetitive tasks, allowing us to focus on high-value work .and record operations without writing a single line of code.

Service now platform expand **Flow Designer** with **Integration Hub** to integrate third party applications **or** services for more comprehensive workflows and automation across your enterprise.

Now, anyone can build flows to deliver business solutions faster. Out-of-the-box core actions and Service Now spokes (scoped applications with Flow Designer content) accelerate flow creation.

Workflow Engine VS. Flow Designer

Workflow Engine is the tried-and-true method that requires a more advanced developer but allows for more complex scenarios, including plenty of readily accessible scripting blocks. We have many core activities for automate our business process on service now application

Vs.

Flow Designer enables users to use natural language to automate tasks, record operations, notifications, and approvals without having to write it in code. It enables various process automation capabilities in a consolidated design environment. Flow Designer lets you build multi-step processes simply by dragging, dropping, and connecting steps.

If you are a beginner or not technical enough to build out processes in code, Flow Designer is going to be your jam. Flow Designer reduces scripting so it can be understood and created easily by the non-technical user. The same end functionality can be tackled by both, but you are going to find Flow Designer a lot easier to use.

Architecture of Flow Designer

We need to know that how flow designer working in servicenow platform
Flow designer may contain varies type of triggers, more actions, subflows and flow logics

Trigger can define when to start our flow, which can be **record based**, **schedule based**, or **application based**

Record-based triggers run a flow after a record has been created, updated, or deleted. The flow can use the triggering record as input for actions.

Schedule-based triggers run a flow at the specified date and time. The flow can use the execution time as input for actions.

Application triggers are added when the associated application is activated.

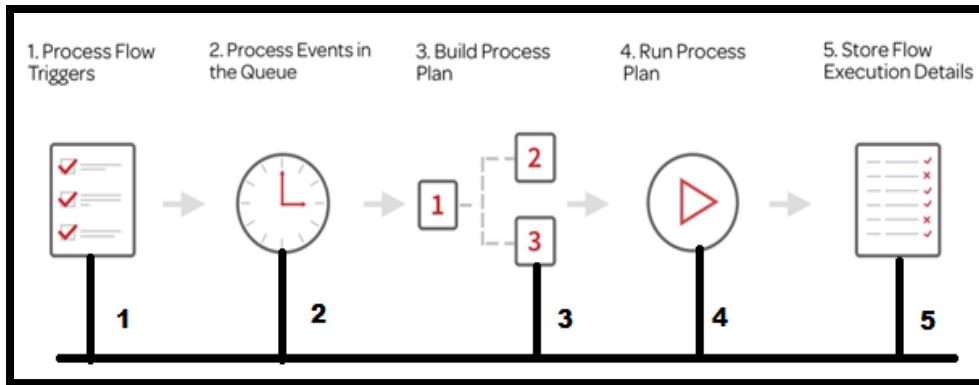
For example, the Metric Base trigger is present when the MetricBase application is active.

Sequence of flow process

- When the flow trigger conditions occur or an API directly calls the flow, the system creates an entry in the event queue to start the flow.
- The scheduler processes the event and starts the flow in the background.
- The system builds a process plan from the flow.
- The system runs the process plan using the record that triggered the flow.
- The system stores the execution details in a context record.

Process Flow Execution diagram

This diagram will give you an idea how the flow is running



Different states in Flow Designer

Flow Designer States	
Complete	The flow is completed successfully
In progress	The flow is running. By default, a transaction quota rule prevents flows from running longer than an hour.
Waiting	The flow is waiting for another event to occur
Cancelled	The flow was cancelled by user
Error	The flow encountered an error and has stopped running.

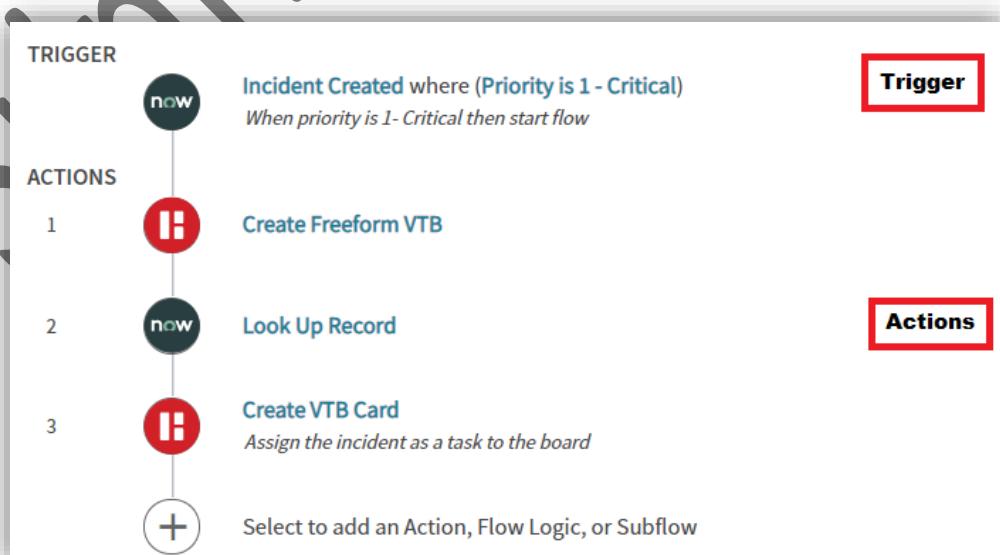
Flow designer Features and Advantages

Flow designer application providing great futures with in application

- Natural Language platform
- Less coding
- Handle complexity data
- Non-technical person also understands and work on it
- Create actions and reusable
- Allowing to call script include code snippets
- Schedule jobs for automate business process
- Included Integration Hub
- Connecting third party application
- Spokes feasibility
- Decouple business policy from processes
- Out-of-the-box flow triggers and action
- Reusability action
- Reduce development costs by providing a library of reusable flow components created by service now developer
- Natural language is used to assist no-code users configure flow components without having to know how to script,
- Process owners and developers can create, operate, and troubleshoot flows from a single interface

What is Flow?

Flow is sequence of actions for automate business process in servicenow platform A flow contains **triggers** and **one or actions**



What is trigger?

The trigger defines the conditions that when start the flow. When our trigger condition is true, the system will start the flow.

Flow designer application can support three types of triggers.

1. Record Based
2. Schedule Based
3. Application based

Record Triggers

Record triggers will be occurring when record was created or updated.

Record Triggers		
Created	Updated	Created or Updated

We have three types of record triggers

Created	Starts the flow when a record created in particular table
Updated	Starts the flow when a record is updated in specific table. But need to select when the will be run
Created or Updated	Starts a flow when a record is either created or updated in a specific table. Requires selecting when to run the flow.

We need to follow below considerations while creating record triggers

1. Trigger condition should be unique on same table
2. Ignore records inserting or updated by import sets or update sets
3. Replace record triggers on Service Catalog tables with Service Catalog application triggers

TRIGGER

Select to abort trigger creation

Trigger Select a Trigger

ACTIONS

Record

- Created
- Updated
- Created or Updated

Created

Trigger initiates from a ServiceNow record creation that meets the condition filter.

Date

- Daily
- Weekly
- Monthly
- Run Once
- Repeat

Application

- Inbound Email

Service Catalog

Date triggers

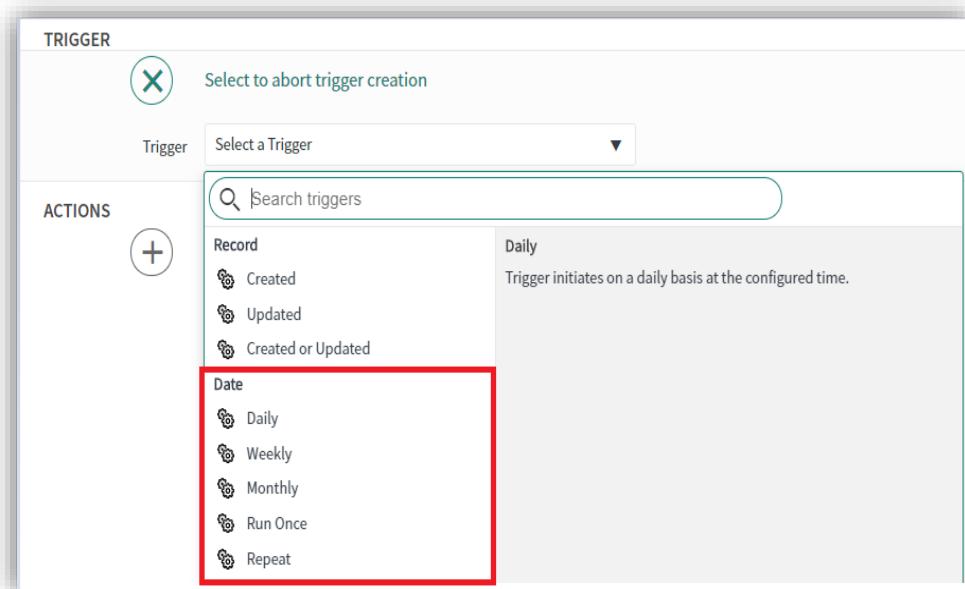
Date triggers are used to start a flow after a specific date and time or repeatedly at scheduled intervals, depend on scheduled you selected

Because flows are processed asynchronously, a flow with a date trigger may not run at the exact scheduled time its trigger conditions were met. For example, if a scheduled flow is triggered during core business hours, the system may have to process other events in the queue before it can run the scheduled flow.

Date Triggers

Daily	Starts a flow at a specific time every day.
Weekly	Starts a flow at a specific time every week.

Monthly	Starts a flow at a specific time every month.
Run Once	Start a flow once at a specific time but does not repeat
Repeat	Starts a flow at regular intervals you define.



Application triggers

Use application triggers to start a flow when application-specific conditions are met.

Application Triggers

SLA Task	Starts a flow when a MetricBase trigger is met. Requires the MetricBase application.
Metric Base	Starts a flow from a Service Catalog item request
Service Catalog	Starts a flow from an SLA definition record
Inbound Email	Starts a flow when our instance receive email

<p>Date</p> <ul style="list-style-type: none"> <input type="radio"/> Daily <input type="radio"/> Weekly <input type="radio"/> Monthly <input type="radio"/> Run Once <input type="radio"/> Repeat <p>Application</p> <ul style="list-style-type: none"> <input type="radio"/> Inbound Email <input type="radio"/> Service Catalog 	<p>Note: Stages are available for all parent Flows. When a Flow is associated with a Service Catalog Item with the Workflow field, the flow stages display when the item is requested. Only stages from the parent flow display. As the Flow progresses, stages on paths that the flow did not take are skipped or removed from the display.</p>
--	--

Flow Designer User Workspace and components

Flow designer user interface is used to create a new flow and managing or modified existing flows



What is Action in Flow?

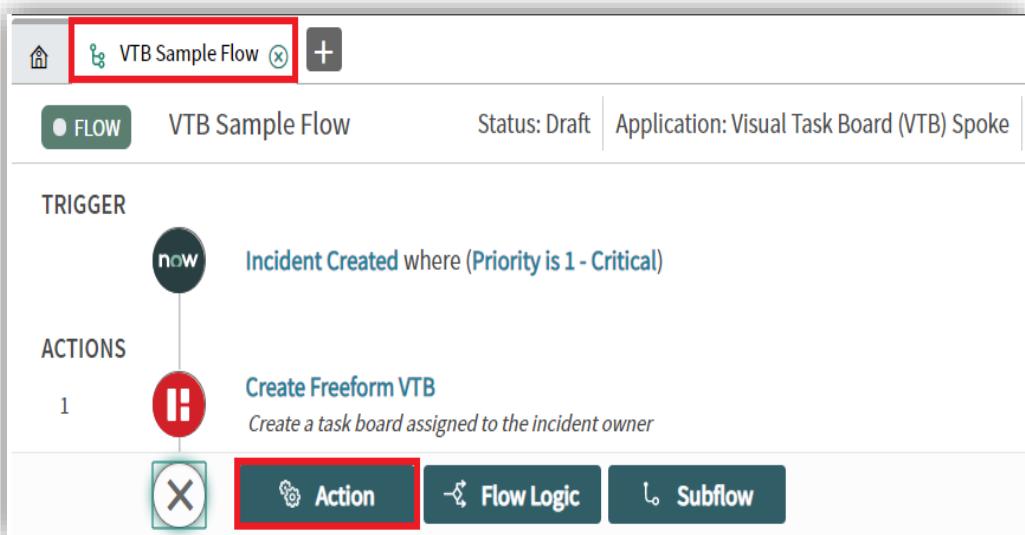
Actions define the work performed by a flow. Now platform included so many system actions

Flow Actions

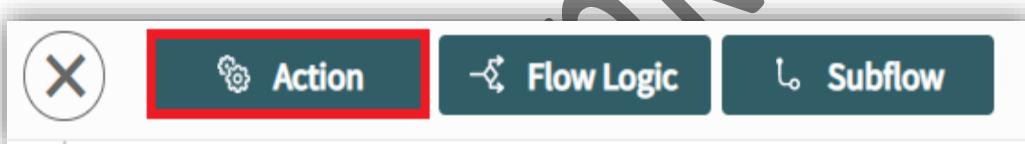
- | | |
|---|---|
| <ul style="list-style-type: none"> ▪ Create Record ▪ Update Record ▪ Delete Record ▪ Create task ▪ Lookup Record ▪ Log ▪ Send Email ▪ Create Catalog Task | <ul style="list-style-type: none"> ▪ Copy Attachment ▪ Delete Attachment ▪ Lookup Attachment ▪ Move Attachment ▪ Create Attachment ▪ Get Catalog Variable ▪ Ask for Approval ▪ Create or Update ▪ Wait For Condition |
|---|---|

How to add actions for your existing flow?

If you want, add an action to existing flow, click the Select to add an Action, Flow Logic, or Subflow link.



Click on Action button to add an action



1. Use the *action* selection pane to select actions to add to a flow.
2. The actions are grouped by application. Within the application,
3. Actions are grouped in categories. For example, the ITSM application has action categories for **Assignment, Change, Incident, Problem, Request, and Miscellaneous**.
4. To see action details, hover over the action.

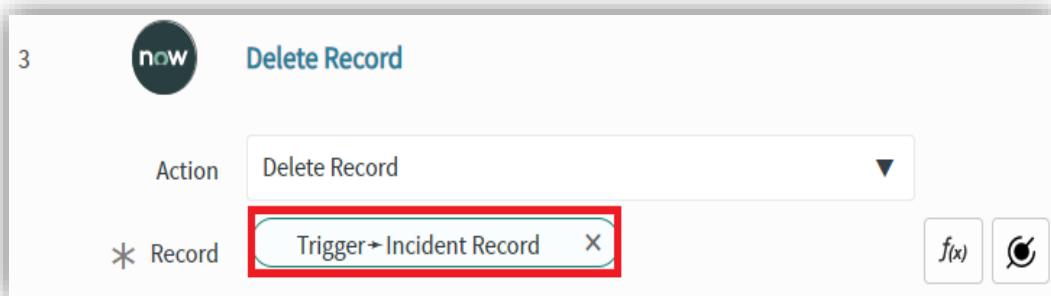
Action Flow Logic Subflow

Search Actions

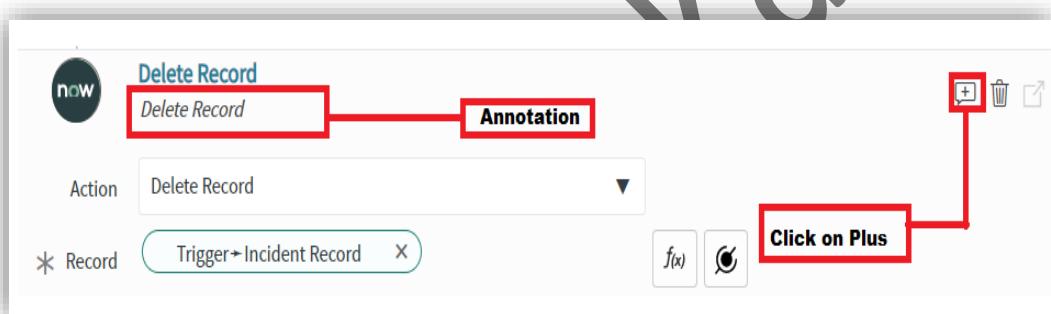
Installed Spokes	Default
now ServiceNow Core	Add Worknote Link to Context
Connect	Ask For Approval
now Continuous Integration and Contin...	Create or Update Record
Global	Create Record
ITSM	Create Task
now Password Reset	Delete Record
Visual Task Board (VTB)	Get Email Header

Every action has a different set of configuration fields. The configuration fields are dependent upon what the action does.

For example, the Delete Record action needs to know the Table in which to create the record and the field values for that record.



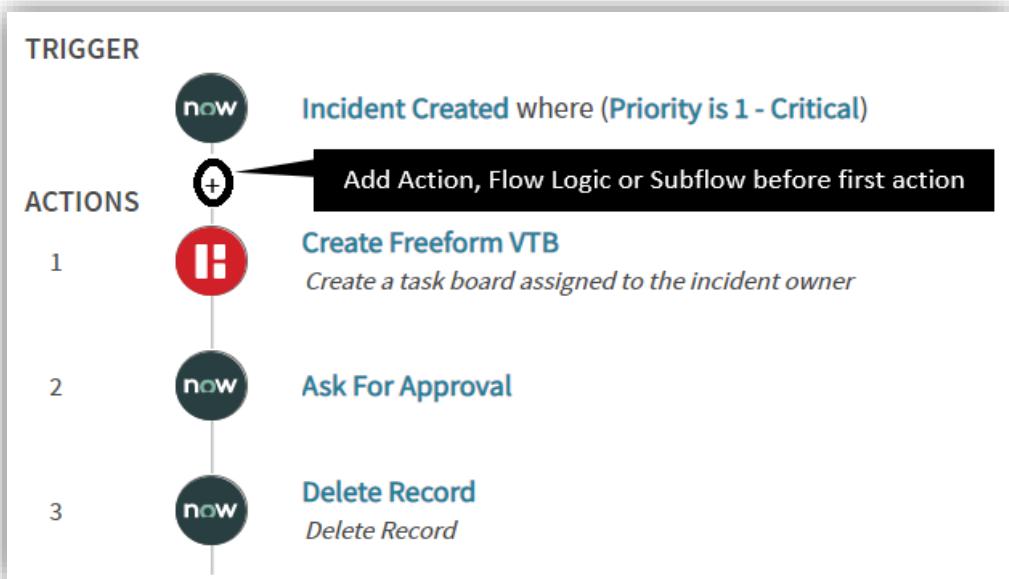
Annotation is defined identify the purpose of an action.



Define Actions in Order

As actions are added, they appear in numbered order in the flow. The **Select to add an Action, Flow Logic, or Subflow** link is available to add new actions to the end of a flow.

To add an action between actions in the flow, hover over the transition line between the actions and click the **Add Action, Flow Logic, or Subflow** button. The tooltip indicates where the action will be added.



NOTE: The **Add Action, Flow Logic, or Subflow** button is only visible when hovering over the transition line.

Flow Designer Roles

Flow Designer is the user interface used to create new flow and modified existing flow and manage all aspects of flows. Users with a **Flow Designer** role or **the admin** role can work with flows. Flow Designer roles are

1. **flow_designer** : Create New Flow and Modified Exist Flow
2. **flow operator**: View flow **execution details, dashboards, and logs**
3. **action_designer** :**Create** and **edit** custom actions

Exercise-1: Working with predefined flow

In this exercise, we will open sample workflow (**VTB Sample flow**)

- First, we can view the **flow's trigger** and **actions**, which are Allready configured
- Then we can execute **the flow** for better understand how it is working.
- Finally, we will see the **results** of the flow.

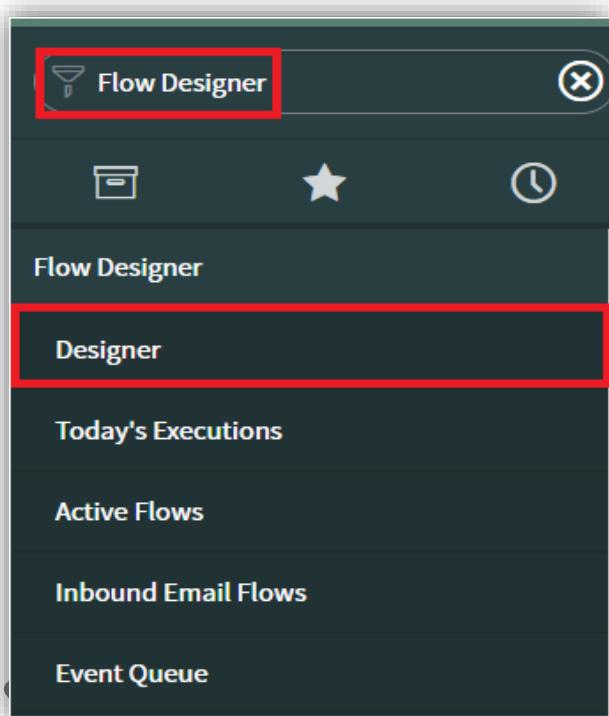
Requirement

In this practice exercise, we will see the **Visual Task Boards (VTB)** currently owned by Beth Anglin. We will see an Incident assigned to Beth Anglin.

When a critical (Priority 1) Incident is created, the **VTB Sample Flow** creates a **Freeform Visual Task Board (VTB)** and creates a card for the Incident in the **to Do lane**.

Procedure

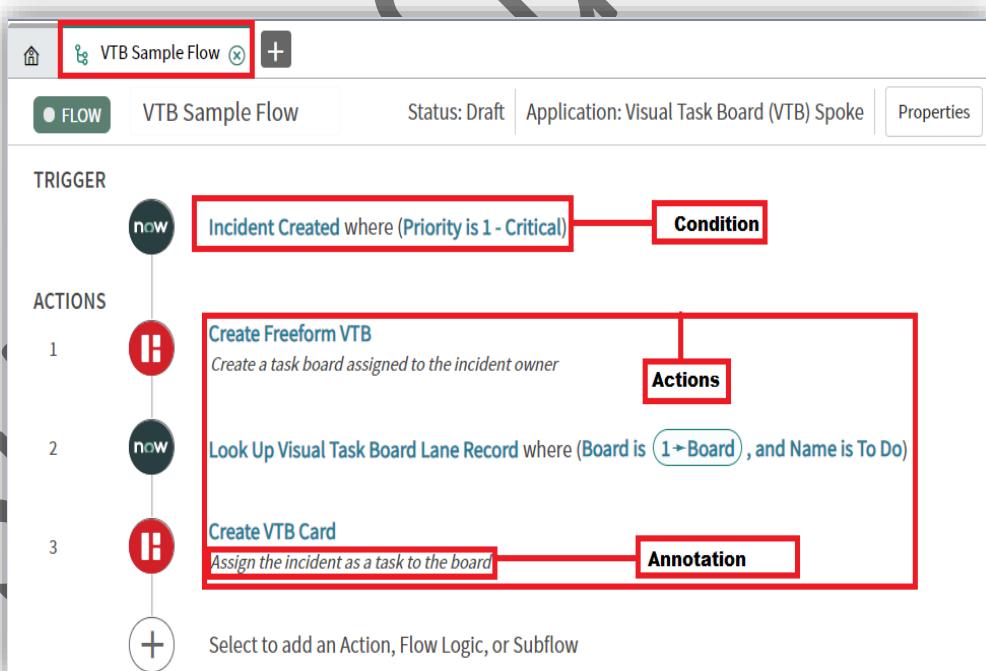
1. Navigate to **Flow Designer** application
2. Click on **Designer** module



3. Open VTB Sample Flow

	VTB Sample Flow	
<input type="checkbox"/>	Recommendation Evaluator	benchmark_recommendation_evaluator Published true
<input type="checkbox"/>	Service Catalog Item Request	service_catalog_item_request Global
<input type="checkbox"/>	SLA notification and escalation flow	sla_notification_and_escalation_flow Global
<input type="checkbox"/>	VTB Sample Flow	vtb_sample_flow Visual Task Board (VTB) Spoke

4. Identify the trigger. The flow triggers when an Incident record is created. What other **conditions** need to be met?
5. Checking the actions. How many **actions** does this flow have?
6. Review each action. What do the actions do?



Execute VTB Sample Flow

1. Impersonate with **Fred Luddy**
2. Check Fred Luddy **Visual Task Board**, it is empty

The screenshot shows the left side of the interface with a dark sidebar containing 'Self-Service' and 'Visual Task Boards' (both highlighted with red boxes). The main area displays a 'Welcome to Visual Task Boards' message with a green circular icon, followed by a list of features and a 'Create New Visual Task Board' button.

3. Create a new **Incident** record and **(Fill all mandatory Fields)**

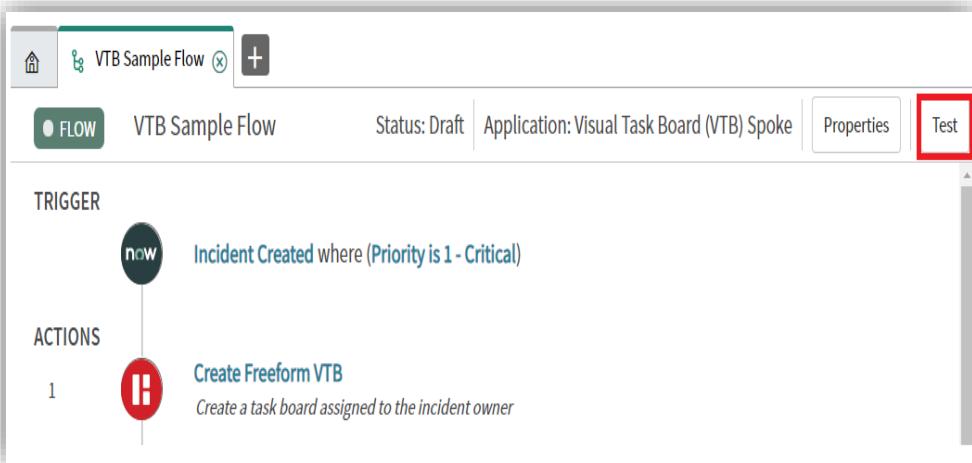
Caller: Abel Tutor, Short Description: Network Issue, Priority: 1-Critical

Assigned to: Fred Luddy

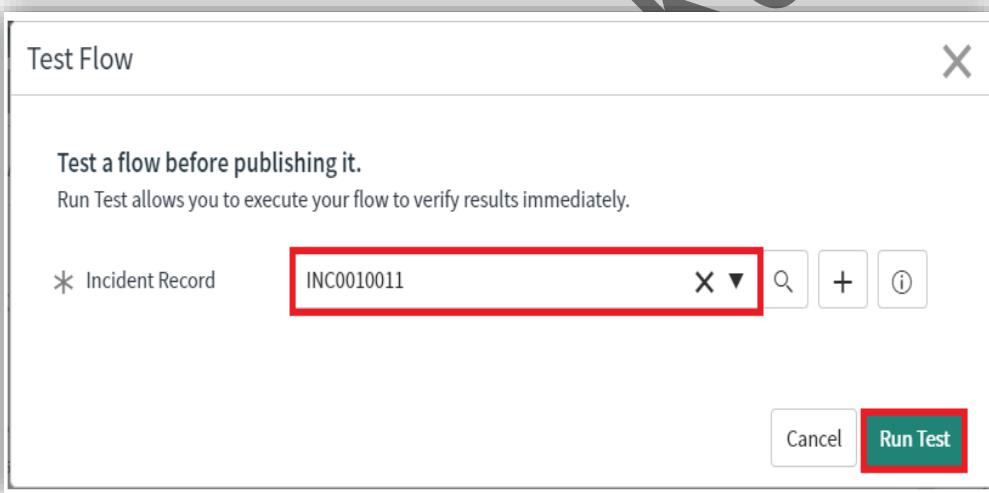
The screenshot shows the 'Create New Incident' form. Fields highlighted with red boxes include:

- Number: INC0010011
- Caller: Abel Tuter
- Category: Inquiry / Help
- Subcategory: -- None --
- Service: (empty)
- Configuration item: (empty)
- Contact type: -- None --
- State: New
- Impact: 1 - High
- Urgency: 1 - High
- Priority: 1 - Critical
- Assignment group: (empty)
- Assigned to: Fred Luddy
- Short description: Network Issue

4. In the **VTB Sample Flow** tab, click the **Test** button.



5. Set the created Incident Record field to **INC0010011**.
6. Click the **Run Test** button.

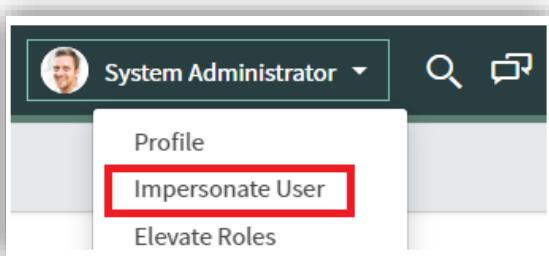


7. When the flow has been executed, click the Close (X) button to close the Test Flow dialog.

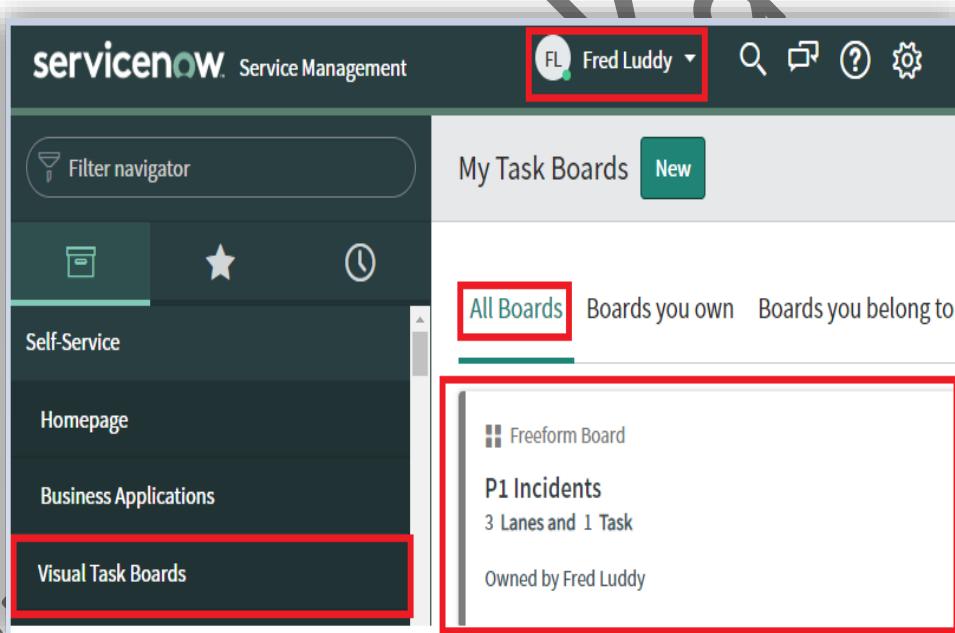
Check Results of VTB Sample Flow

Let see results how the **VTB sample flow** added Visual T Board to **Fred Luddy**

1. Impersonate with **Fred Luddy**



2. Navigate to **Self-Service**→**Visual Task Board**



3. Open **P1-incident** and check the details



4. The Look Up action in the **VTB Sample Flow** finds the to Do lane in the new VTB and adds the Incident to the **to Do lane**.

Exercise-2: Critical-1 Incident Flow

Requirement:

When customer created **Priority-1** incident (**Flow Will Start**)

Trigger Condition

Incident Created where (priority is **1-Critical**, and active is true)

Actions

1. If Incidents short description value contain '**Issue with networking**' (**If Condition**)
2. Auto assign the value to assignment group is **Network** and Incident **Caller Manger** add as a **Watch List (Update Incident Record)**
3. Add one work note link to context(**To see execution details**)
4. Send Email to **Caller** about incident creation
 - **To:** Incident Caller
 - **CC:** Caller Manager
 - **Subject:** New P1 Incident created by: Caller Name
 - **Body :** P1 Incident Reference number: Incident Number

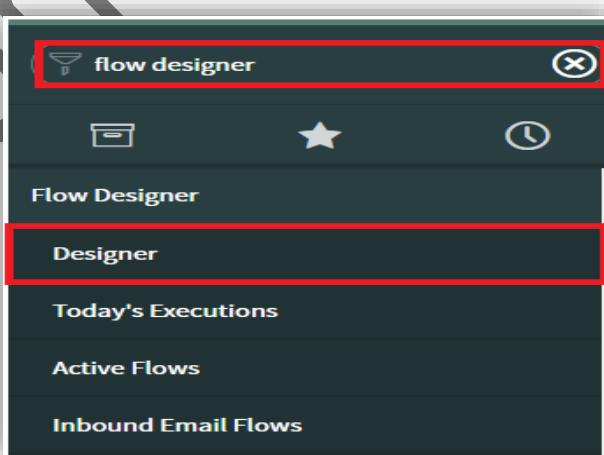
Assignment group: Incident Assigned Group

Incident Assigned To: Incident Assigned to

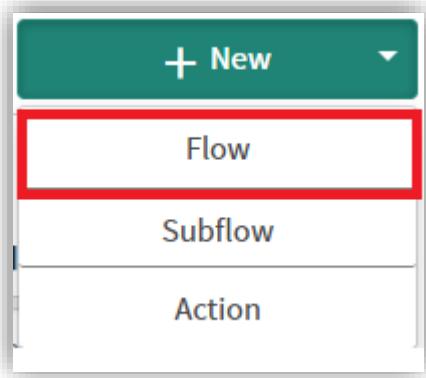
5. Add **assigned group users** to task conversation

Procedure

1. Navigate to **Flow designer**→**Designer**



2. Click on **Designer**



3. Click on **Flow** (To create New Flow)
4. Fill all required details for flow record

Flow Properties

Name: Critical-1 Incident Flow

Application: Global

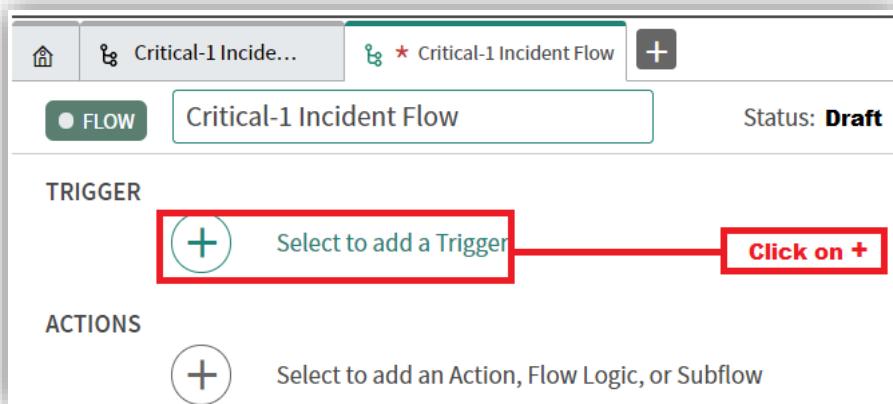
Description: When we created a Priority-1 incident (Flow Will Start)

Run As: User who initiates session

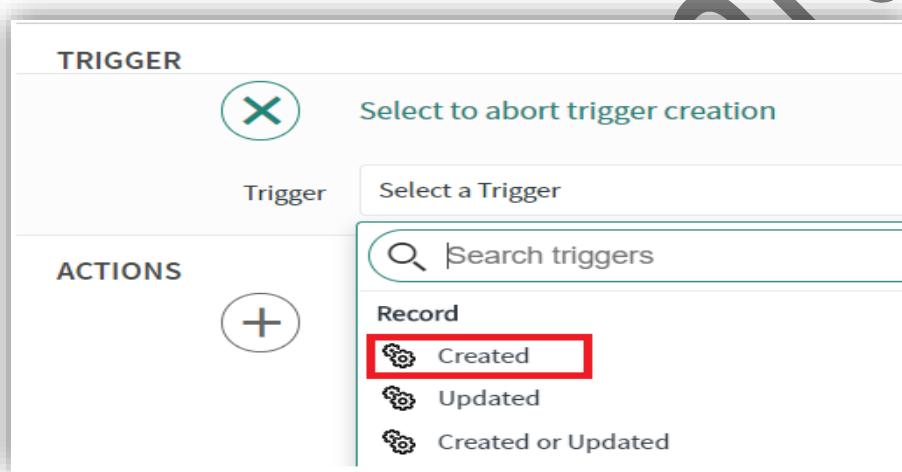
Protection: -- None --

Cancel Submit

5. Name: Critical-1Incident Flow
6. Application : Global
7. Description: When we created a **Priority-1** incident (**Flow Will Start**)
8. Click on **Submit**



9. Click on + to add **Trigger Condition**



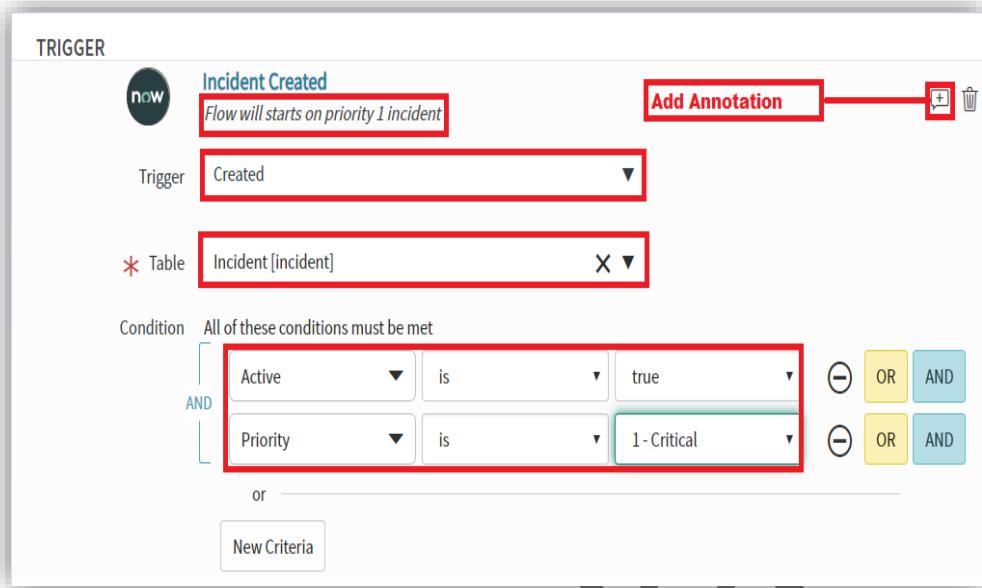
10. **Annotation:** Flow will start on priority 1 incident

11. **Trigger:** Created

12. **Table :** incident

13. **Condition :** Active is true and priority is 1-critical

14. Click on Done



15. After created trigger condition it will show look like below screen shot

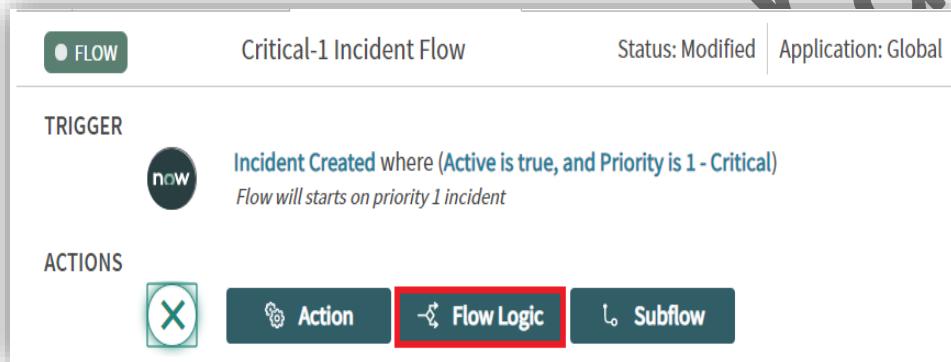
The screenshot shows the configuration of a flow named 'Copy of Critical-1 Incident Flow'. It has a status of 'Modified'. The trigger section shows 'Incident Created where (Active is true, and Priority is 1 - Critical)'. The actions section is currently empty, showing a placeholder 'Select to add an Action, Flow Logic, or Subflow'.

Create Actions

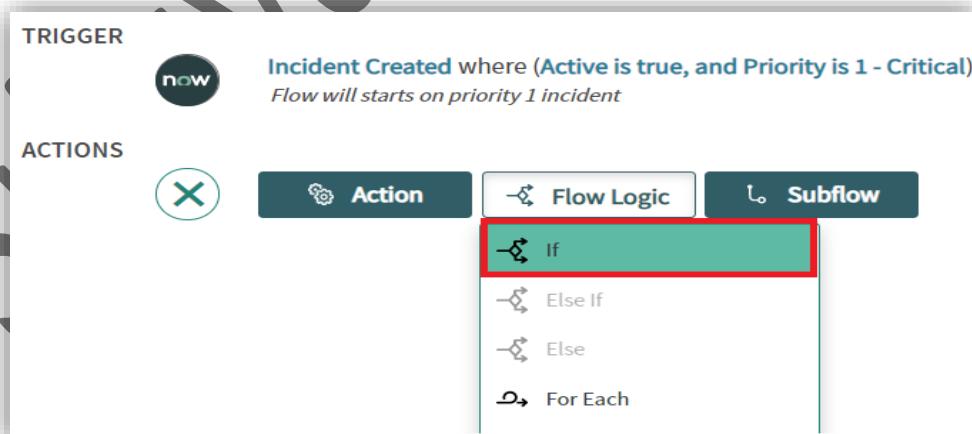
Create multiple actions for your flow to automate your business process

Procedure

1. Open your **Priority-1 Incident Flow**
2. Click on **Action + icon**
3. Click on **Flow Logic** button



4. Select If statement
5. Fill the If condition record (Check condition to If Evaluated true then perform next operation)

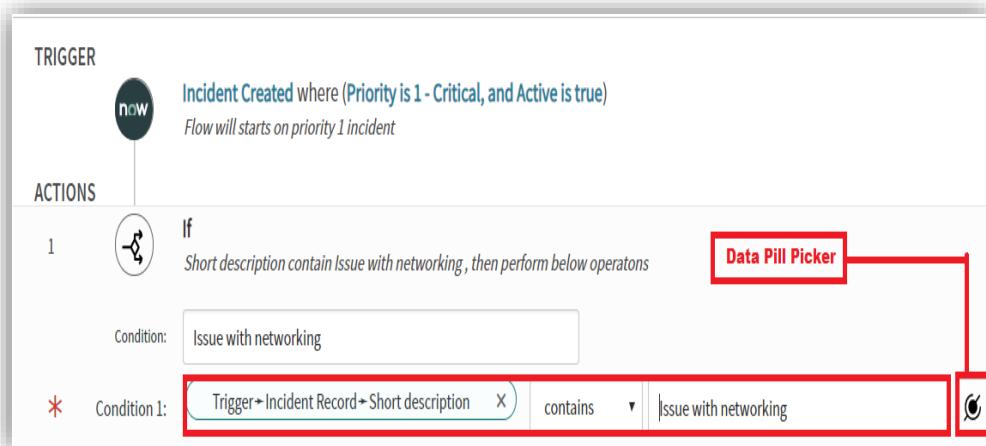


6. **Annotation:** Short description contain Issue with networking, then perform below operations
7. **Condition :** Issue with networking
8. **Condition1 :** Trigger→Incident Record→Short Description Contains Issue with networking

Trigger -> Incident Record -> Short description X

is

v Issue with networking



9. Use Data pill to fill condition 1

trigger - Record Created -> Incident Record -> short_description

Trigger - Record Created	Incident Record	Record
	Incident Table	Table
	Run Start Time	Date/Time
	Short description	String
	State	Choice
	Subcategory	Choice

```
graph LR; T1[Trigger - Record Created] --> IR[Incident Record]; IR --> SD[short_description];
```

10. Click on + icon Add more actions to your flow

TRIGGER
now
Incident Created where (Priority is 1 - Critical, and Active is true)
Flow will starts on priority 1 incident

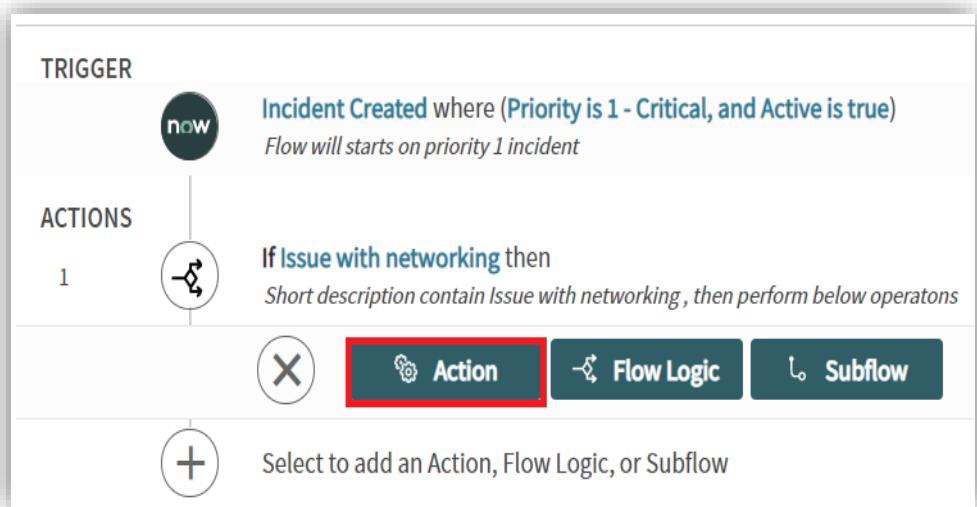
ACTIONS

1 If Issue with networking then
Short description contain Issue with networking, then perform below operators

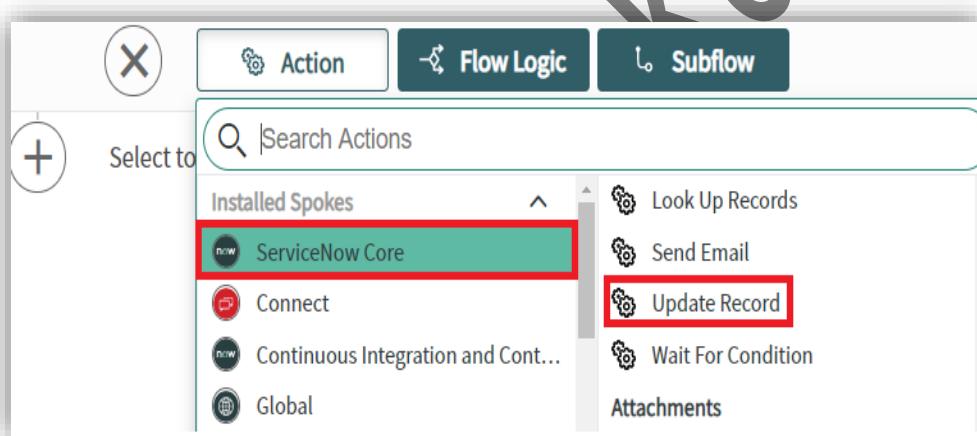
+ Select to add an Action, Flow Logic, or Subflow

```
graph TD; subgraph Trigger [Trigger]; direction TB; T1[Incident Created where Priority is 1 - Critical, and Active is true]; T2[Flow will starts on priority 1 incident]; end; subgraph Actions [Actions]; direction TB; A1{If Issue with networking then Short description contain Issue with networking, then perform below operators}; A2[+ Select to add an Action, Flow Logic, or Subflow]; end; A1 --> A2;
```

11. Click on Action button



12. Fill action form



13. Action: Update Record

14. Record: Trigger→Incident Record

15. Assignment group: Network

16. Watch List: Trigger→Incident

Record→Caller→Manager→Name

17. Click on Done

now Update Incident Record

Action: Update Record

* Record: Trigger->Incident Record

* Table: Incident [incident]

* Fields: Assignment group, Network

Watch list: Trigger->Incident Record-> Caller-> Manager-> Name

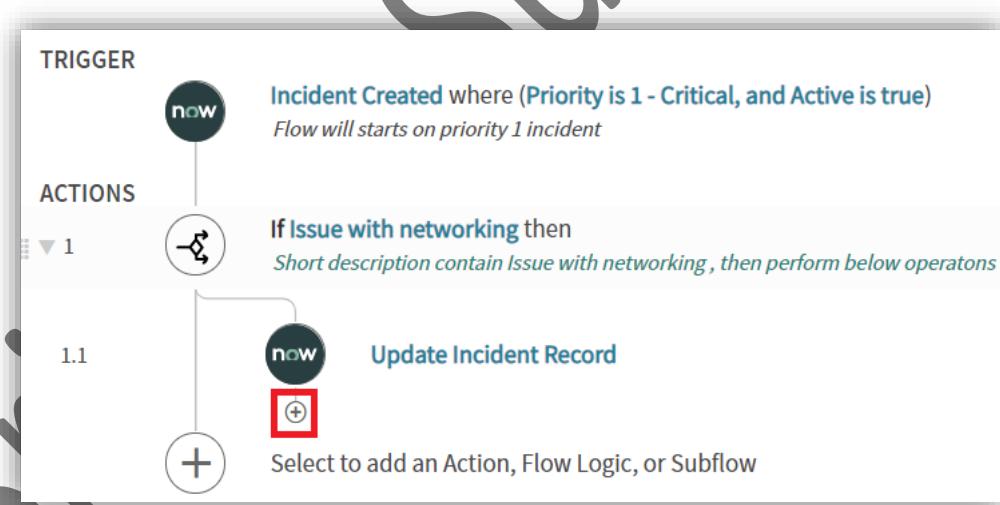
+ Add Field Value

Done

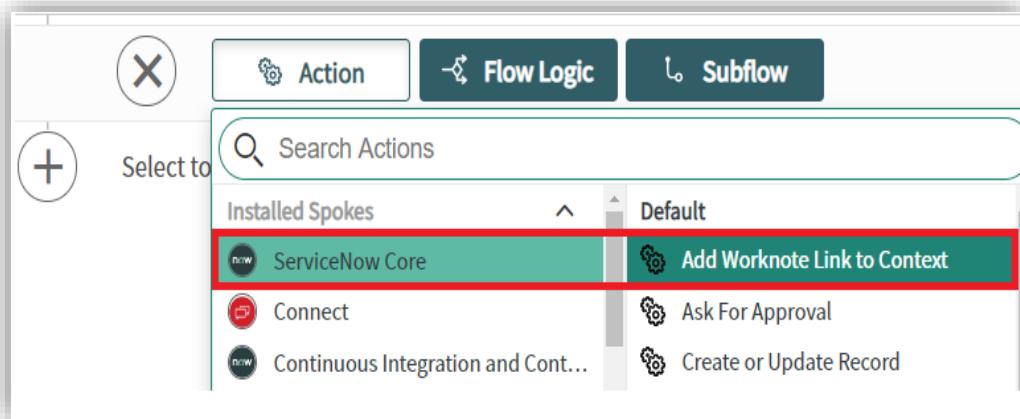
18. Add work note link to context

19. Click on + icon

20. Create new action



21. Fill Add worknote link to context form



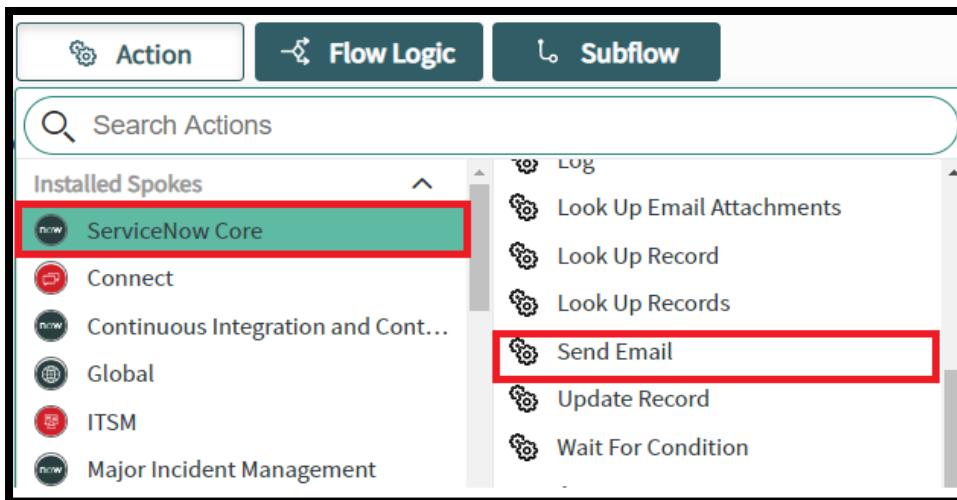
22. **Action :** Add work note link to context
23. **Table:** Incident
24. **Record:** Trigger→Incident Record
25. **Journal Field:** Work notes
26. **Additional Comment:** New P1 Incident created by:
Trigger→Incident Record→Caller→Name
27. Click on **Done**

1.2 Add Worknote Link to Context

Action	Add Worknote Link to Context
* Table	Incident[incident]
* Record	Trigger→Incident Record
* Journal Field	Work notes
Additional Comments	Priority-1 Flow started

Delete Cancel Done

28. Add **Send Email** Action
29. Click on **+** Icon
30. Click on **New Action**



31. Fill Send Email action Form

The screenshot shows the configuration form for the 'Send Email' action. The 'Action' field is set to 'Send Email'. The 'Target Record' is set to 'Trigger-> Incident Record'. The 'Table' is set to 'Incident [incident]'. The 'Include Watermark' checkbox is checked. The 'To' field contains the path 'Trigger-> Incident Record-> Caller-> Email'. The 'CC' field contains the path 'Trigger-> Incident Record-> Caller-> Manager-> Email'. The 'Subject' field contains the text 'New P1 Incident created by behalf of you:' followed by the path 'Trigger-> Incident Record-> Caller-> Name'. The 'Body' section contains three fields: 'P1 Incident Reference number:' with the path 'Trigger-> Incident Record-> Number', 'Assignment group:' with the path 'Trigger-> Incident Record-> Assignment group-> Name', and 'Incident Assigned To' with the path 'Trigger-> Incident Record-> Assigned to-> Name'.

Action	Send Email
Target Record	Trigger-> Incident Record
Table	Incident [incident]
Include Watermark	<input checked="" type="checkbox"/>
To	Trigger-> Incident Record-> Caller-> Email
CC	Trigger-> Incident Record-> Caller-> Manager-> Email
Subject	New P1 Incident created by behalf of you: Trigger-> Incident Record-> Caller-> Name
Body	P1 Incident Reference number: Trigger-> Incident Record-> Number Assignment group: Trigger-> Incident Record-> Assignment group-> Name Incident Assigned To: Trigger-> Incident Record-> Assigned to-> Name

32. Action: Send Email
33. Target Record: Trigger→Incident Record
34. Include Watermark: Checked
35. Table: Incident
36. To: Trigger→Incident Record→Caller→Email
37. CC: Trigger→Incident Record→Caller→Manager→Email
38. Subject: New P1 Incident created by behalf of you
Trigger→Incident Record→Caller→Name

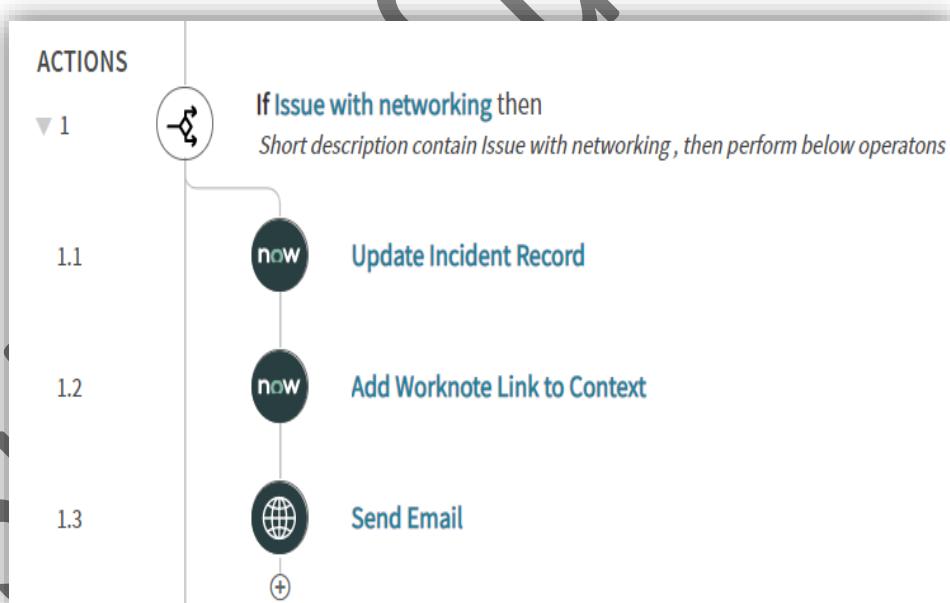
39. Body

P1 Incident Reference number: Trigger→Incident Record→Number

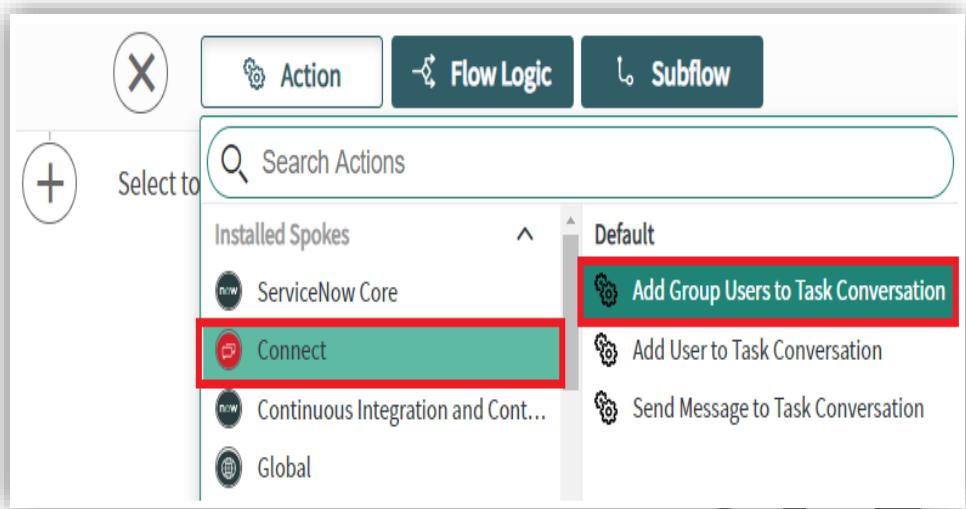
Assignment group: Trigger→Incident Record→Assignment group→Name

Incident Assigned To : Trigger→Incident Record→Assigned to→Name

40. Click on Done



41. Add Group Users to Task Conversation Action
42. Click on + Icon
43. Click on New Action

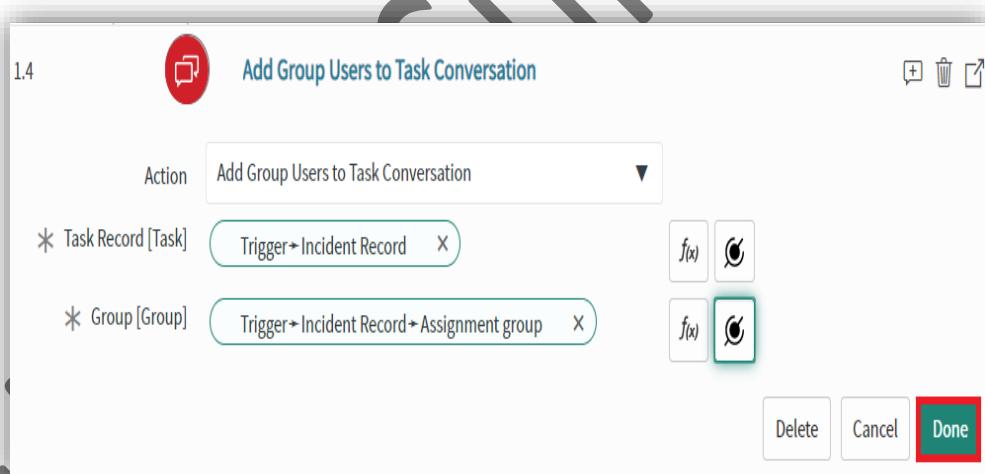


44. Fill Group Users to Task Conversation action Form

45. Action: Group Users to Task Conversation

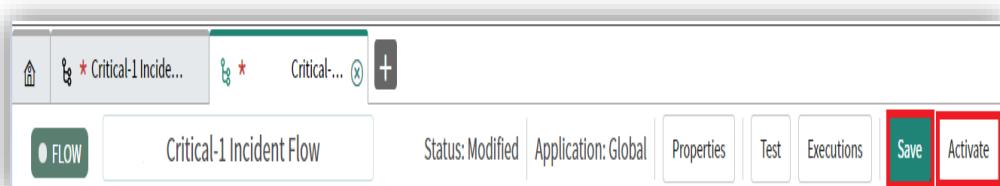
46. Task Record[Task]:Trigger→Incident

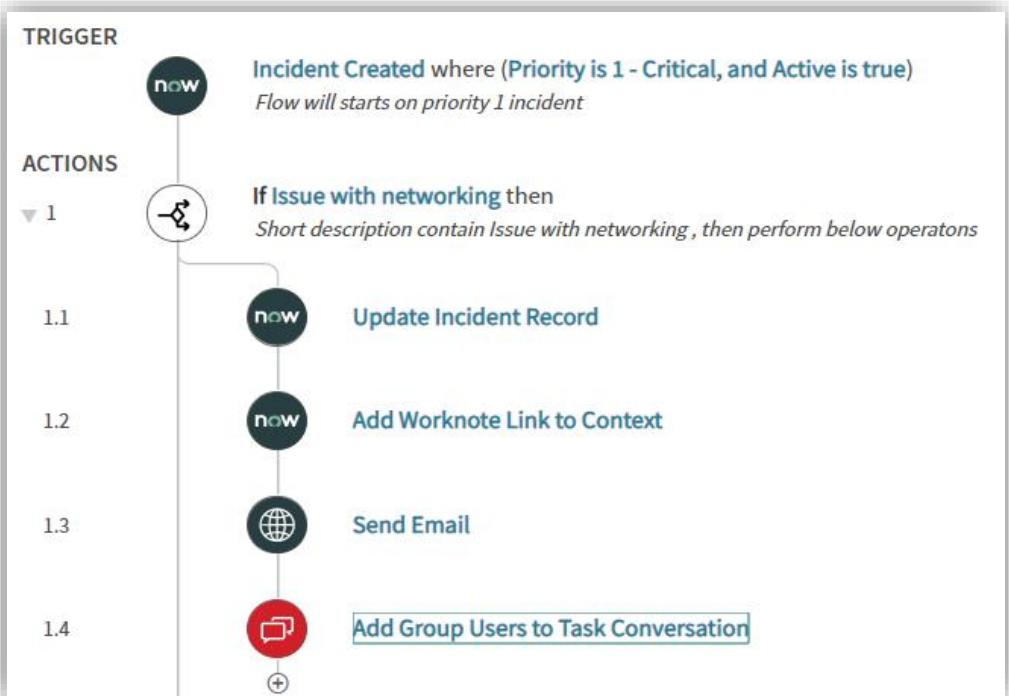
47. Group→ Trigger→Incident→Assignment Group



48. Click on Done

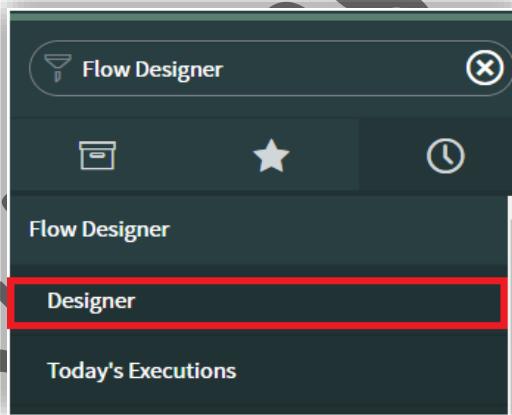
49. After Created Flow successfully then Save and Activate your flow





Test Your Flow

1. Navigate to → Flow Designer
2. Click on Designer Module



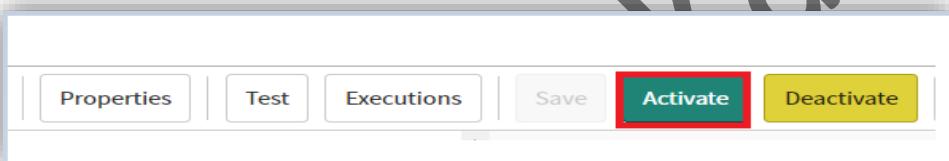
3. Click on Flows tab (Display list flows)
4. Open Your flow which is implemented by you

Flows Subflows Actions Executions Help

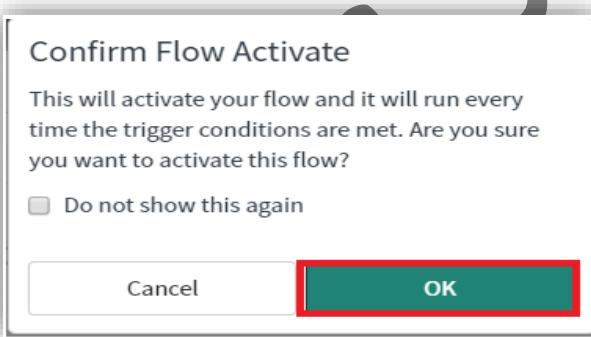
All

	Name	Internal name	Application	Status
<input type="checkbox"/>	Benchmark Recommendation Evaluator	benchmark_recommendation_evaluator	Benchmarks Spoke	Published
<input type="checkbox"/>	Critical-1 Incident Flow	gautham_test	Global	Draft

5. After open your flow click on **Activate** button (activate your flow)



6. Open dialog window clicks on **Ok** button



7. Click on **Test** button

Critical-1 Incide...

FLOW Critical-1 Incident Flow Status: Published Application: Global Properties

TRIGGER

now Incident Created where (Priority is 1- Critical, and Active is true)
Flow will starts on priority 1 incident

8. Click + and Create New Incident

9. Fill all required details

The form contains the following fields:

- Number: INC0010060
- Contact type: -- None --
- * Caller: Abel Tuter
- Category: Inquiry / Help
- Major incident state: -- None --
- Subcategory: -- None --
- Impact: 1 - High
- Service: (empty)
- Urgency: 2 - Medium
- Configuration item: (empty)
- Priority: 2 - High
- Assignment group: (empty)
- Assigned to: (empty)
- * Short description: Issue with networking

10. Click on Submit

The dialog box has the following content:

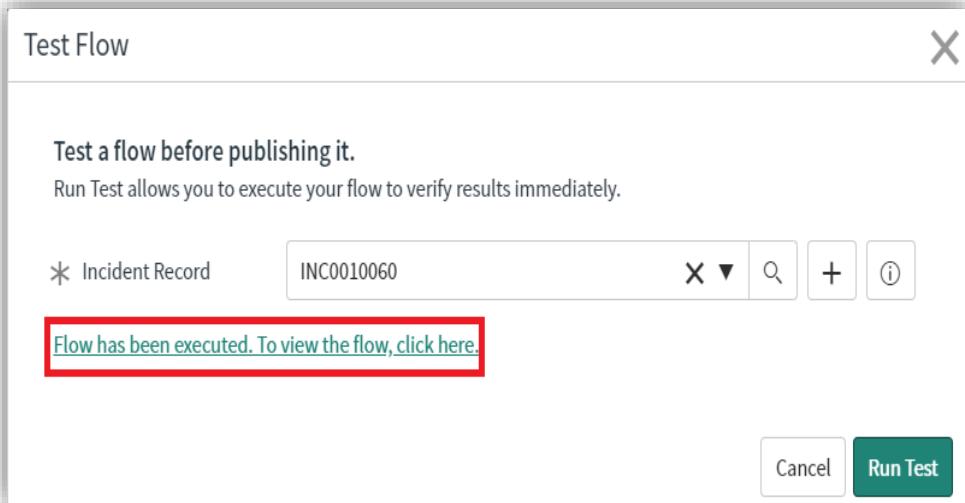
Test Flow

Test a flow before publishing it.
Run Test allows you to execute your flow to verify results immediately.

* Incident Record: INC0010060

Buttons: Cancel, Run Test

11. Select your test record and click on Run Test button



- 12.** Click on **Execution Details** link to see execution history of your flow

	State	Start time	Duration
Incident Created	Completed	2020-05-02 04:54:25	2065ms
If Issue with networking then	Evaluated - True	2020-05-02 04:54:25	2053ms
Update Record	Completed	2020-05-02 04:54:25	1760ms
Add Worknote Link to Context	Completed	2020-05-02 04:54:27	144ms
Send Email	Completed	2020-05-02 04:54:27	33ms
Add Group Users to Task Conversation	Completed	2020-05-02 04:54:27	116ms

- 13.** If you want check each action details as expanded

VARIABLE NAME	TYPE	CONFIGURATION	RUNTIME VALUE
Task Record	Reference	Trigger + Incident Record	INC0010060 0
Group	Reference	Trigger + Incident Record + Assignment group	

No Logs

▶ Steps

Exercise-3:

Incident state changes to on-hold and hold reason is awaiting change then the flow will start

Requirement

When Incident record state update with **on-hold** and **on hold reason** is **awaiting change** then this flow will start

Trigger Condition

Incident updated where (state change to **3**, and on hold reason is Awaiting Change)

Actions

1. Ask for Approval

CAB Department need to approve for create new change

2. If CAB Person Approved then

At least 2 members need to approved

3. Create a change request record with following fields

- Configuration Item
- State
- Short Description
- Assignment Group
- Assigned To
- Category

4. Update Incident Record with below fields

- State is **closed**
- Work notes
- Resolution Code
- Resolution Note

5. Log Action

Store a log file into system for better understand

6. Send Email

- **To:** Incident Record Assigned group mail
- **CC:** Incident record assigned to
- **Subject:** New change record was created for this incident
- **Body:** When CAB manager approved, Change request will be created automatically

7. Slack Post Message

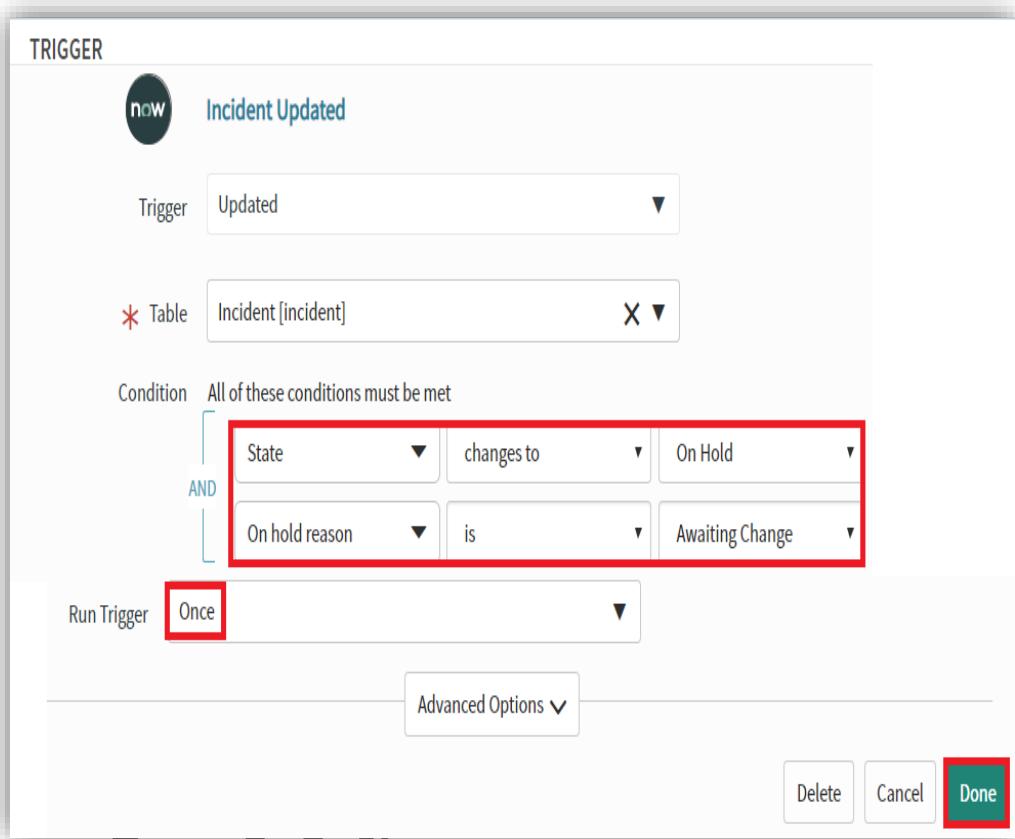
- **Web hook URL:**
<https://hooks.slack.com/services/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>
- **Message :**Defined your message
- **Username :** Servicenow

8. Post Incident Details Microsoft Team Message

- **Your Webhook URL**
<https://outlook.office.com/webhook/000d77d7-4fc8-47ea-951a-d869ba2ae2e9@78c0a43d-bc3b-47ff-b313-95e3f31c56c2/incomingWebhook/706018012d4b44319aa81884ee57779d/7a12d0fd-03d4-4e27-b4fe-4c55efdfa7b0>

- **Title:** New change record is created
- **Additional Message:** Working with new change record

1. Create Trigger Condition
2. Click on + Select add to a Trigger
3. Click on Done



4. Add Ask For Approval Action

ACTIONS

1 now Ask For Approval

Action: Ask For Approval

* Record: Trigger=Incident Record

Table: Incident[incident]

Approval Field: Approval

Journal Field: Approval history

* Rules

Approve When:

1 # of users approve CAB Approval

Remove rule set

Add another OR rule set

Remove rule set

OR

Reject When:

1 # of users reject CAB Approval

Remove rule set

OR AND -

Delete Cancel Done

5. Click on **Done**
 6. Add **Flow Logic**

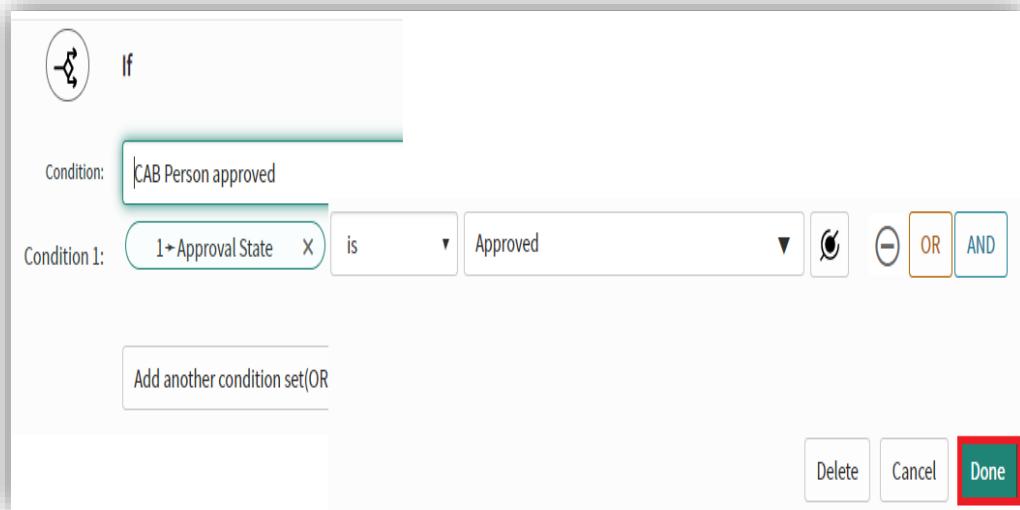
If

Condition: CAB Person approved

Condition 1: 1+ Approval State X is Approved ▾ OR AND

Add another condition set(OR)

Delete Cancel Done



7. Click on **Done**

8. Add action **Create Record**

now Create Change Request Record

Action Create Record ▾

* Table Change Request [change_request] X f(x) C

* Fields Configuration item X Trigger> Incident Record> Configuration item X

State X New ▾

Assignment group X Trigger> Incident Record> Assignment group X

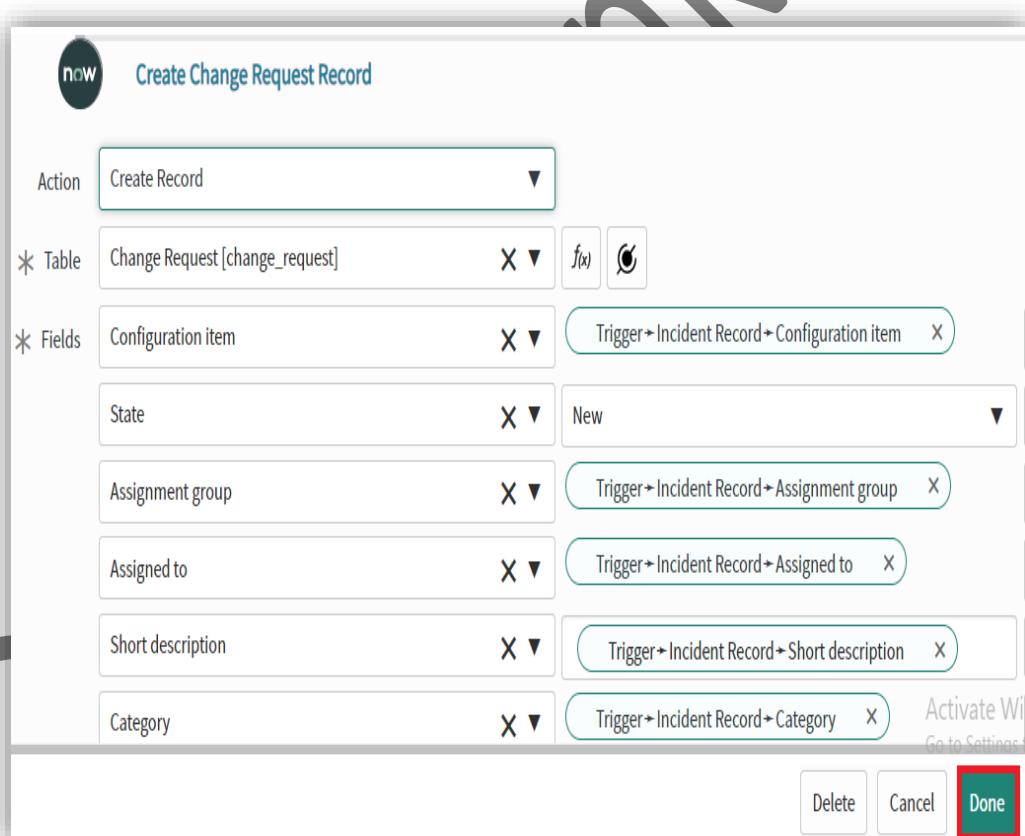
Assigned to X Trigger> Incident Record> Assigned to X

Short description X Trigger> Incident Record> Short description X

Category X Trigger> Incident Record> Category X

Activate Wi Go to Settings

Delete Cancel Done



9. Click on **Done**

10. Add **Update Record** Action

now Update Incident Record

Action: Update Record

* Record: Trigger->Incident Record X f(x) S

* Table: Incident [incident] X ▼ f(x) S

* Fields: State X ▼ Closed ▼

Work notes X ▼ New change record was created for this incident and find change number
2.1-> Change Request Record-> Number X

Resolution code X ▼ Solved (Work Around) ▼

Resolution notes X ▼ New change record was created for this incident and find change number
2.1-> Change Request Record-> Number X

Delete Cancel Done

11. Click on Done

12. Add Log Action

now Log

Action: Log

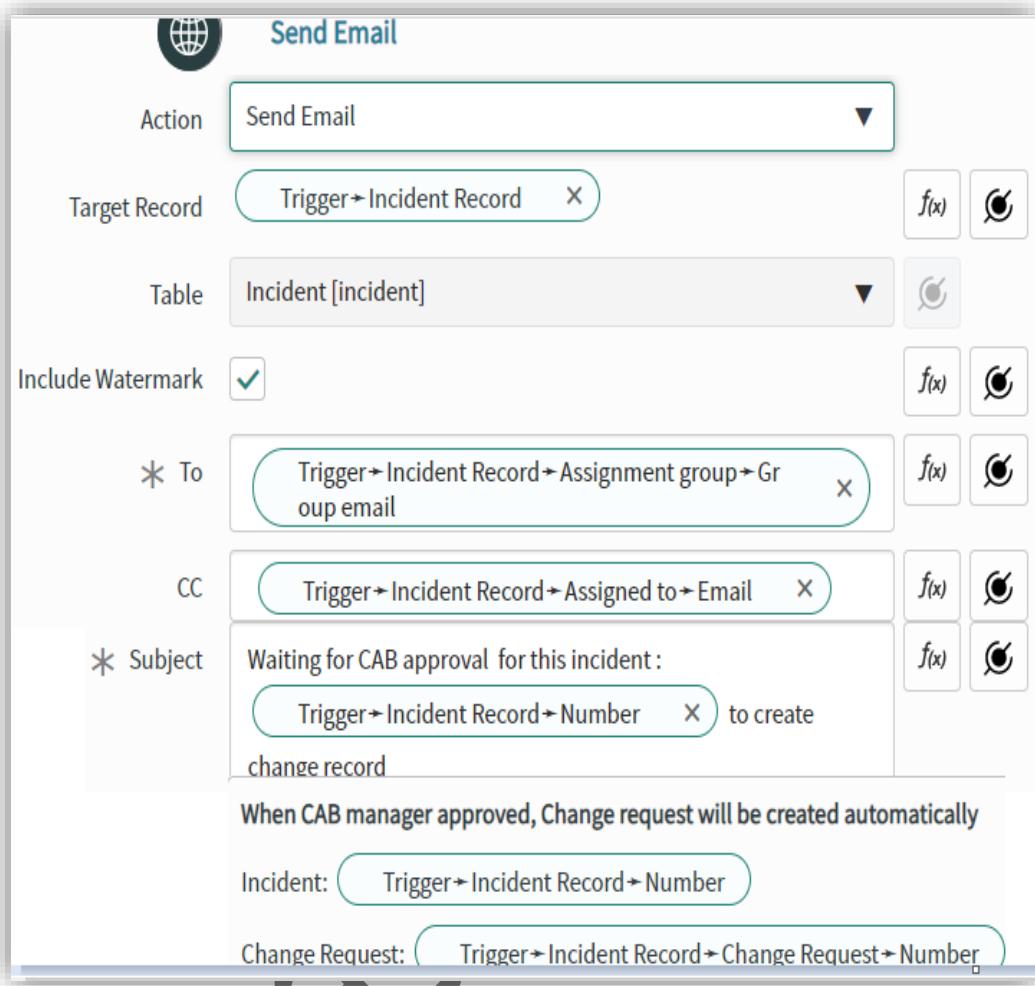
Level: Info

Message: New change record was created from
Trigger->Incident Record X

Delete Cancel Done

13. Click on Done

14. Add Send Email Action



15. Click on Done
16. Add Post Message action from Slack spoke (Internal Communicator)

Post a Message

Action: Post a Message

Webhook URL: <https://hooks.slack.com/services/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>

Message:

- Created a new change request for this incident:
Trigger → Incident Record → Number
- Find CR number for Reference:
Trigger → Incident Record → Change Request → Number

Username: Servicenow

Channel: Servcenow

Icon: :heart:

17. Click on Done

Post Incident Details

Action: Post Incident Details

Webhook URL: <https://outlook.office.com/webhook/000d77d7-4fc8-47ea-951ad869ba2ae2e9@78c0a43d-bc3b-47ff-b313-95e3f31c56c2/IncomingWebhook/706018012d4b44319aa81884ee57779d/7a12d0fd-03d4-4e27-b4fe-4c55efd7b0>

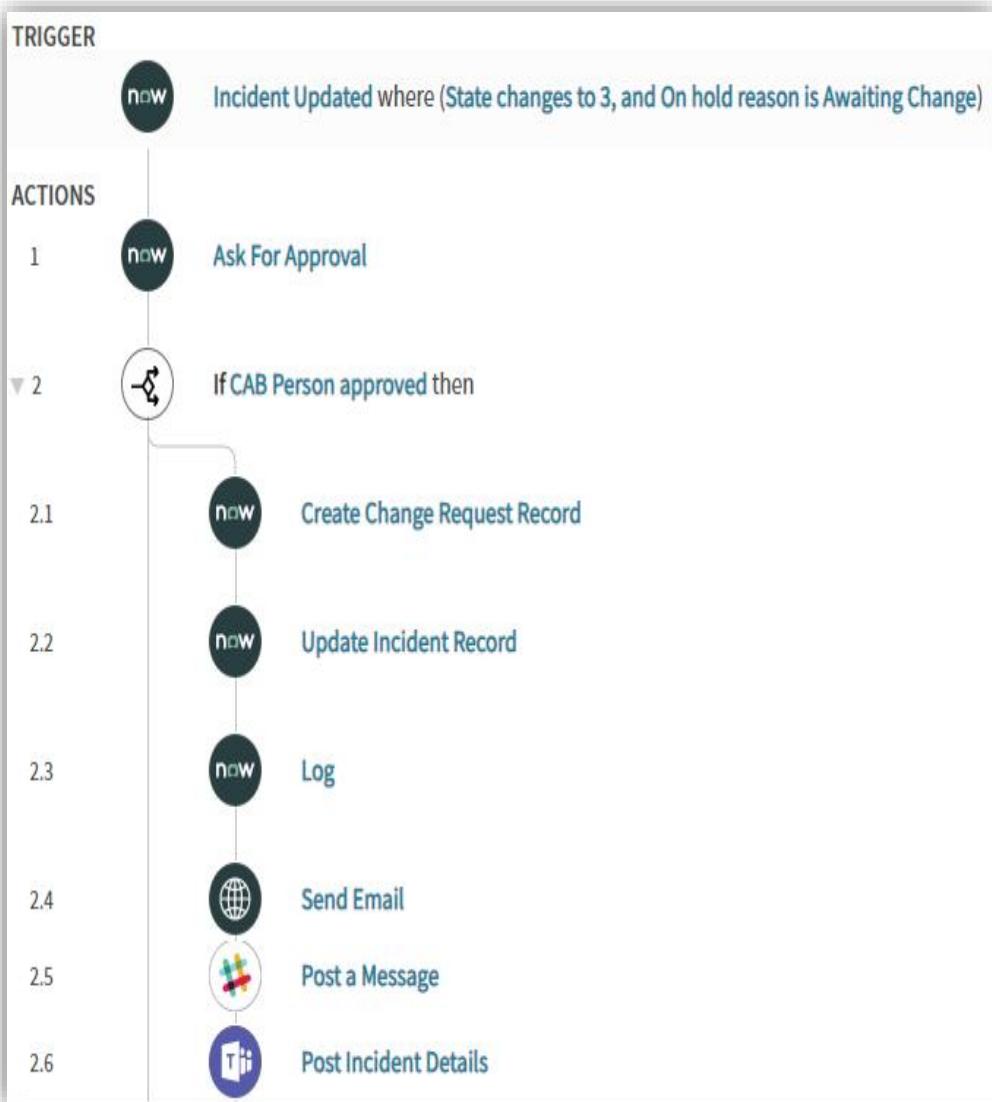
Incident [Incident]: Trigger → Incident Record

Title: New change record is created

Additional Message: Working with new change record

18. Click on Done

19. Get Flow Designer Condition and action Diagram below



Srinivas

Exercise-4: Create a flow with service catalog item

If you want work with service catalog item flow process, need to activate the **Flow Designer support for the Service Catalog** plugin (com.glideapp.servicecatalog.flow_designer). This plugin activates related plugins if they are not already active.

Requirement

When customer requested for add a role to user through **Service Catalog** item Then this this flow will start

Trigger Condition

Select **Service Catalog** trigger

Actions

1. Get catalog variables

Get catalog variables from catalog item template

2. Ask for Approval

Ask approval for requester manager

3. If Condition

If manger approved the request (**Move to next activity**)

4. Create User role record

Create user role record in [**sys_user_has_role**] table

5. Else Condition

If manager rejected the request, then rejected and create log information about reject reason

Procedure

1. Create a catalog item to run our flow

2. Navigate to **Service Catalog** → **Catalog Definition** → **Maintain Item**

3. Click on **New**

4. Fill Catalog Item form

The screenshot shows the 'Service Catalog' interface for creating a new catalog item. The 'Name' field is filled with 'Request a role for user'. The 'Catalogs' dropdown is set to 'Service Catalog'. The 'Category' field contains 'Can We Help You?'. The 'Application' is 'Global' and 'Active'. At the bottom, the 'Item Details' tab is selected, while other tabs like 'Process Engine', 'Picture', 'Pricing', and 'Portal Settings' are visible but not selected.

5. Create two variables into your catalog item

The screenshot shows the 'Variables' section of the Service Catalog. A red box highlights the 'Variables (2)' tab. Below it, there are tabs for 'Variable Sets', 'Catalog UI Policies', 'Catalog Client Scripts', and 'Available For'. Under 'Available For', there are links for 'Catalogs (1)', 'Related Articles', and 'Related Catalog Items'. The main area shows a table with two rows. The first row contains a checkbox, a blue info icon, the text 'Reference', and the label 'Select a role'. The second row contains a checkbox, a blue info icon, the text 'Reference', and the label 'Selected role for user'. A red box highlights the entire table row.

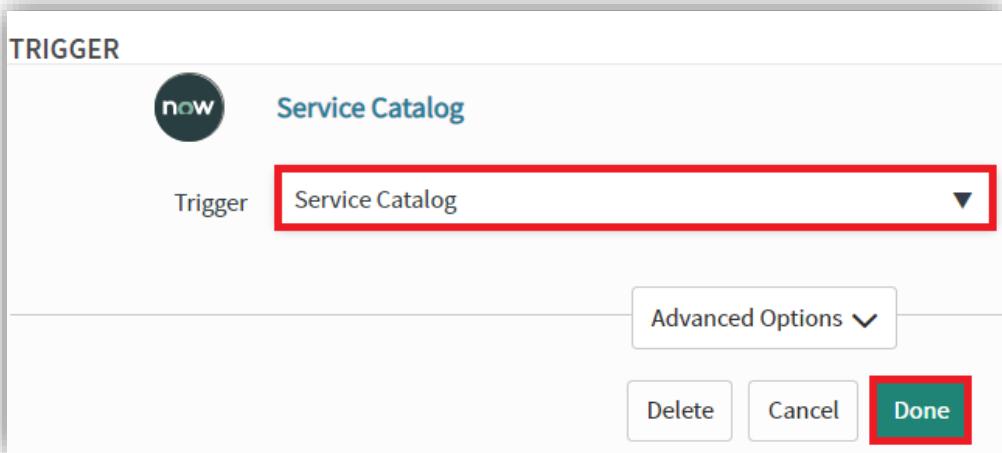
6. Save the record

Create a flow for this service catalog item

1. Navigate to **Flow Designer** → **Designer**
2. Click on **New flow**
3. Fill flow **Properties** form

The screenshot shows the 'Flow Properties' dialog box. It has fields for 'Name' (set to 'Add a role for user'), 'Protection' (set to '--None--'), 'Application' (set to 'Global'), 'Description' (set to 'Add a role for the user'), and 'Run As' (set to 'User who initiates session'). At the bottom right are 'Cancel' and 'Submit' buttons, with 'Submit' being highlighted by a red box. A watermark 'SSSunkar' is visible across the dialog.

7. Click on **Submit**
8. Add Trigger Condition



9. Click on Done

10. Add more actions to our flow

11. Add Get Catalog Variables to get catalog item variables

The screenshot shows the configuration for the 'Get Catalog Variables' action. At the top left is a 'now' icon followed by the action name 'Get Catalog Variables' and a description 'from Request a role for user'. Below this is a dropdown menu labeled 'Action' with 'Get Catalog Variables' selected, which is highlighted with a red box. To the right of the dropdown is a 'Submitted Request [Requested Item]' field with a green border and a 'Trigger → Requested Item Record' link. Further down is a section titled 'Use the following to generate outputs for use in the flow'. It includes a 'Template Catalog Item [Catalog Item]' field with 'Request a role for user' selected, which is highlighted with a red box. Below this are two columns: 'Catalog Variables Available' (containing 'No available values') and 'Selected' (containing 'select_role' and 'selected_role_for_use'). In the bottom right corner are three buttons: 'Delete', 'Cancel', and a green 'Done' button, which is also highlighted with a red box.

12. Click on Done

13. Add Ask for Approval Action

2 Ask For Approval

Action	Ask For Approval
* Record	Trigger → Requested Item Record
Table	Requested Item [sc_req_item]
Approval Field	Approval
Journal Field	Approval history
* Rules	
Approve When: Anyone approves 1 → selected_role_for_user → Manager	
OR	
Reject When: Anyone rejects 1 → selected_role_for_user → Manager	

14. Click on Done

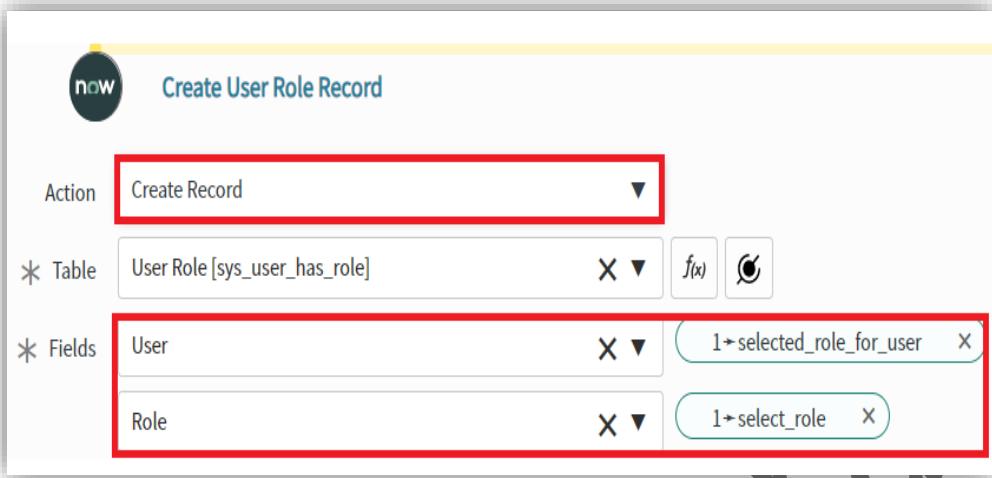
15. Add If Flow Logic

3 If

Condition:	Required manager approval
* Condition 1:	2 → Approval State is Approved
Add another condition set(OR)	
<input type="button" value="Delete"/> <input type="button" value="Cancel"/> <input style="background-color: red; color: white; border: 2px solid red; padding: 2px; margin-right: 10px;" type="button" value="Done"/>	

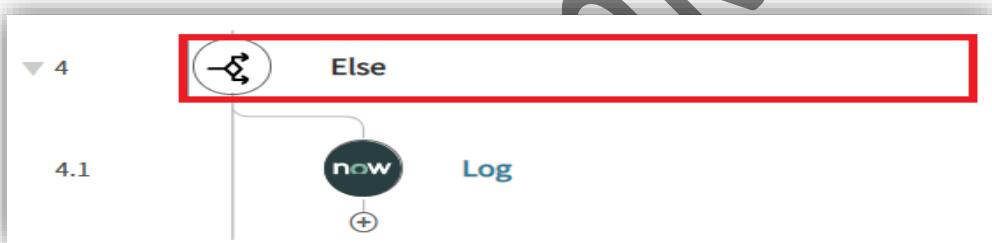
16. Click on Done

17. Add Create User Role Record Action

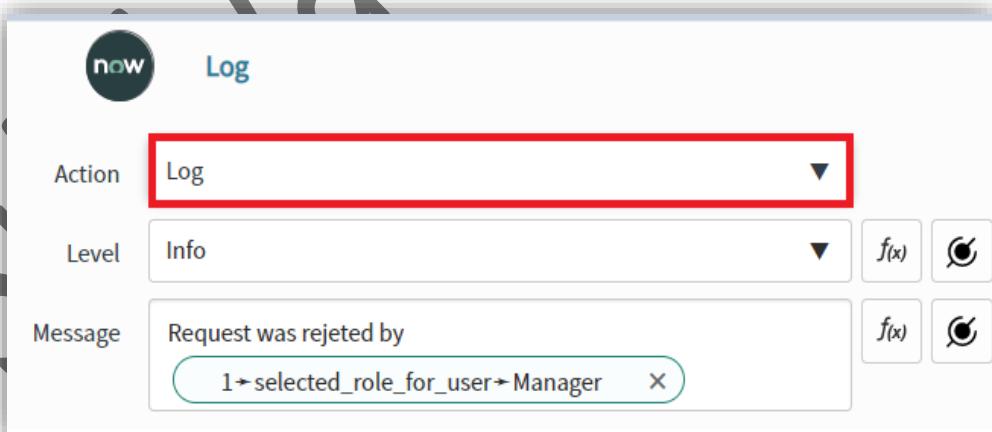


18. Click on Done

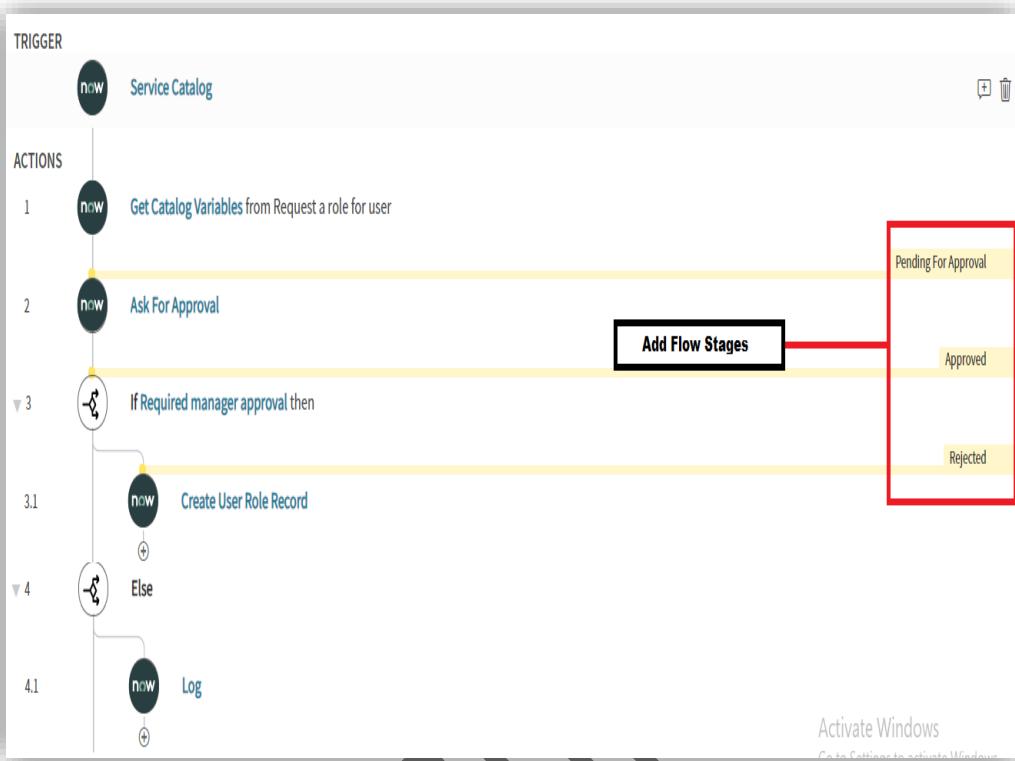
19. Add Else Condition Flow Logic



20. Add Log Action



21. See Flow design diagram below



Add this Flow to our Catalog Item

22. Navigate to Service Catalog → Catalog Definition → Maintain Item
23. Open Request a role for user Catalog Item
24. Go to Process Engine Tab
25. Add our flow

The screenshot shows the "Request a role for user" Catalog Item details. The "Name" field is "Request a role for user". The "Application" is "Global". The "Catalogs" section shows "Service Catalog" is selected. The "Active" checkbox is checked. The "Category" is "Can We Help You?". The "Process Engine" tab is selected, highlighted with a red box. Below it, a note says: "Select the appropriate process engine for the catalog item. Only one engine can be selected." Under the "Flow" section, "Add a role for user" is selected and highlighted with a red box. The "Workflow" section is empty.

26. Save catalog Item record
27. Click on Try it button in the same item
28. Fill the two variables and Click on Order Now Button

This record producer used to request a role for user

* Select a role

atf_ws_designer	<input type="button" value="Search"/>	<input type="button" value="Info"/>
-----------------	---------------------------------------	-------------------------------------

Selected role for user

Abel Tuter	<input type="button" value="Search"/>	<input type="button" value="Info"/>
------------	---------------------------------------	-------------------------------------

Order this Item
 Quantity: 1
 Delivery time: 0 Days

 Shopping Cart
 Empty

Test Our Flow

1. Navigate to Flow Designer → Designer
2. Open Flow Tab
3. Open the Flow

Add a role for user

Flows Subflows Actions Executions Help

All

	Name	Internal name	Application
<input type="checkbox"/>	Add a role for user	add_a_role_for_user	Global

4. Save and Active Flow

Application: Global	<input type="button" value="Properties"/>	<input type="button" value="Test"/>	<input type="button" value="Executions"/>	<input style="background-color: #009640; color: white; font-weight: bold; border: none; padding: 5px; width: 100%; height: 30px;" type="button" value="Save"/>	<input type="button" value="Activate"/>	<input style="background-color: #FFD700; border: none; padding: 5px; width: 100%; height: 30px;" type="button" value="Deactivate"/>
---------------------	---	-------------------------------------	---	--	---	---

5. Click on **Executions**
6. Check our **Flow Execution**
7. It is showing **waiting** for manager approval

The screenshot shows the "EXECUTION DETAILS" page for a flow named "Add a role for user". The top right corner indicates the flow is "Waiting". Below this, the "State" column for the second action is highlighted with a red box and labeled "Waiting".

Stages:	State
(1) (2) (3)	Waiting

FLOW STATISTICS: Executed as: System Administrator | Open Flow Logs

TRIGGER: Catalog Item Requested

ACTIONS:

- 1 now Get Catalog Variables from Request a role for user | Core Action | Completed
- 2 now Ask For Approval | Core Action | Waiting

8. If the manager is **approved the request**, then Roles will be add to user record
9. And Flow will be **Completed**

The screenshot shows the "EXECUTION DETAILS" page for the same flow. All stages are now marked as "Completed". The "State" column for the third action is highlighted with a red box and labeled "Completed".

Stages:	State	Start time	Duration
(1) (2) (3)	Completed	2020-05-07 18:12:29	1266ms

FLOW STATISTICS: Open Flow Logs

TRIGGER: Catalog Item Requested

ACTIONS:

- 1 now Get Catalog Variables from Request a role for user | Core Action | Completed | 2020-05-07 18:12:29 | 23ms
- 2 now Ask For Approval | Core Action | Completed | 2020-05-07 18:12:29 | 257ms
- 3 now If Required manager approval then | Flow Logic | Evaluated - True | 2020-05-07 18:17:48 | 218ms
- 3.1 now Create Record | Core Action | Completed | 2020-05-07 18:17:48 | Activate Windows | 172ms

Interview Questions

1. If you want to include an SLA timer in your process use

Ans: Workflow

2. If you have new business process you would like to automate use

Ans: Flow Designer

3. If your instance is on Kingstone or newer release use work flow
or flow designer

Ans: Flow Designer

4. If you want modify existing logic already developed using
workflow use

Ans: Workflow

5. If you don't want use script, then go for Workflow or Flow
Designer

Ans: Flow Designer

6. Automated sequence of action that run each time when
conditions are meet

Ans: Flow

7. Must be evaluate true for a flow to execute

Ans: Trigger Condition

8. Reusable operations provide by service now in the base system

Ans: Core Actions

9. Role that grants access to flow designer operations information

Ans: flow_operator

10. Role that grant users access to Flow Designer

Ans: flow_designer

11. View run time information about a flow directly in the design
environment

Ans: Execution Details

- 12. Created every time an action is added to a flow and stores variables the action generates**

Ans: Data Pills

- 13. A sequence of reusable actions that can be started from a flow, subflow, or script**

Ans: Subflow

- 14. Required by a Flow but not Subflow**

Ans: Trigger

- 15. It is good practice to create subflow with this type of application**

Ans: Scoped Application

- 16. Passes data between the subflow and a flow, subflow, or script**

Ans: Input and Output

- 17. Stores the available data from each section that can be used by the subflow**

Ans: Data Panel

- 18. The section can include Actions, Flow Logic, or Other subflows**

Ans: Actions

- 19. Flow Logic that populates the output with data**

Ans: Assigned Subflow Output

- 20. Controls if and when an action or subflow is executed**

Ans: Flow Logic

- 21. When a series of conditions are met the action(s) below the condition are executed**

Ans: If

- 22. Optional conditional statement to execute actions when an if statement is false**

Ans: Else if/Else

- 23. Flow logic that executes actions within this statement on every record in the list**

Ans: For Each

- 24.** Repeatedly apply one or more actions until the condition is meet

Ans: Do the following until

- 25.** Execute multiple action at the same time

Ans: Do the following parallel

- 26.** Compare input values to a decision table to determine the actions(s) to execute

Ans: Make a decision

- 27.** Set the amount of time the follow should pause. A duration and schedule are required

Ans: Wait for a duration of time

- 28.** Workflow that are active and published can be called flow designer

Ans: Call a Workflow

- 29.** Used within an If, Else if or Else to stop a flow/subflow

Ans: End

- 30.** Specify what is returned in the subflows output

Ans: Assign Subflow Output

- 31.** Role that enables a user to launch the action design environment to create and edit actions

Ans: action_designer

- 32.** Reusable operation that enable process analysts to automate Now Platform features

Ans: Action

- 33.** A container for any combination of actions, Flow Logic, and Subflows

Ans: Action Section

- 34.** By default, flows are set to use this number of actions per flow

Ans: 50 Actions

- 35.** Service now provided action available to any flow subflow

Ans: Core Action

- 36. Provides a way to create functionality that is not available in the base instance**

Ans: Custom Action

- 37. Roles required to use the action designer**

Ans: action_designer or Admin

- 38. This grouping of reusable operations makes them easier to find and maintain**

Ans: Spokes

- 39. A single reusable operation in the Action Designer that carries out the work**

Ans: Action Step

- 40. Name 3 characteristics of a flow**

Ans

- Performs the same predefined process every time it executes
- Automates business logic for an application or process
- An Automated sequence of action that run each time condition is meet

- 41. A trigger specifies when a flow should execute**

Ans: True

- 42. Name 2 benefits of testing flows in the design environment**

Ans

- Triggers are simulated which automatically begin the execution of the flow,
- Multiple test can be run against the same test record

- 43. Name 3 Characteristics of an action**

Ans

- Reusable operations for automating system features,
- Automates process outside of a Service now instance
- Enables Execution of third part communication APIs

44. Name 3 Trigger Types

Ans: Record based, Schedule-Based, Application

45. What are subflow inputs responsible for

Ans: They specify the data available to the subflow when it launches

46. When a flow is processed the system stores the execution

details in a record in the Log[syslog]table

Ans: False

47. Name 2 roles grant full access to all flow designer features in every application scope

Ans: flow_designer, admin

48. A single reusable operation within an action is known as an

Ans: Action Step

49. 2 reasons why it is considered a good practise to always create Flow Designer content within a scoped application vs. The 'Global' scope

Ans

- Scoped helps categorize content and makes it easier to maintain and release
- Scope protects an application and its artifacts from damage to, or from other application

50. Name 3 components of an Action

Ans: Step, Output, Input

51. Name of the run trigger option that must be selected for the flow to execute every time the trigger condition is met

Ans: For each unique change

52. What should you do if you find yourself repeatedly configuring actions with same configuration settings

Ans: Create custom action pre-set with the configurations required

53. Name 3 benefits of the Flow Designer

Ans:

- Reduce development costs by providing a library of reusable flow components created by service now developer
- Natural language is used to assist no-code users configure flow components without having to know how to script,
- Process owners and developers can create, operate, and troubleshoot flows from a single interface

54. For a flow to be available for execution in the instance, it must

be

Ans: Published

55. What are action outputs

Ans: Data Variables used within the action

56. An action specifies what the flow should execute

Ans: True

57. What occurs every time an action is added to a flow

Ans: Data Pills are created to capture runtime variables

58. What is the Action Designer used for?

Ans: Enables a user to launch the design environment to create and edit actions

59. Name 2 characteristics of a subflow

Ans:

- Contains input and output that pass data to and from the subflow.
- Sequences reusable actions that can be started from a flow, a subflow, or a script

60. What is main purpose of flow logic

Ans: It is the programming structure to make decisions, branch, and make logical choices based on user input

61. Why are subflow properties important

Ans: They can set a protection policy as well specify the runs as setting

Srinivas Sunkara

Lesson-14

Knowledge Management Life Cycle and State Model

Agenda

- Knowledge Management Overview
- Available Roles in Knowledge Management
- Knowledge Management Usage and Benefits
- Knowledge Management Architecture
- Knowledge Management Life Cycle and State Model
- Describe key knowledge management personas
- Workflow of a Knowledge Article
- Knowledge Management Properties and Features
- Knowledge Management Article Versioning
- Describe the article quality index and how to execute the check list
- Have an understanding of the available management dashboards, reports, home pages in service now platform
- Use of Knowledge article template

Sri

Knowledge Management Overview

Knowledge management is an application with in service now platform.

The service now knowledge management application is used to sharing of information as knowledge base. These all knowledge bases may contain number of knowledge articles. This application will support the process of Knowledge management and sharing the documents in our business unit.

Information becomes knowledge once it is documented, actionable and can be shared and consumed. The Knowledge Management Service Portal enables users to access a portal view of knowledge bases and articles. It is available by default for new customers. Knowledge management allows create and categorize knowledge article as per business requirement. Knowledge management will gather all knowledge articles which are what user read the information.

Knowledge management classified into

- User Criteria for KB base
- Provide troubleshooting tips
- Maintain company rules and policies
- Process Documents
- FAQs
- Questions and Answers
- Technical Solutions
- Categorized Articles
- Getting Feedback about each KB article
- Self Service Opportunities
- Workflow for Published and Retired of our KB article
- All users can search for knowledge on the Service Portal
- Provide articles in Incident form
 - Articles are organized by category
 - Describe the purpose and goals of knowledge management
 - Increase knowledge relevancy by empowering agents and employees to capture knowledge articles in context during work processes, such as case or incident management.

Knowledge Management Roles

KM Roles	
Knowledge	Access Knowledge Application menu and Articles
knowledge_manager	Knowledge managers perform administrative functions for knowledge bases they manage, such as defining categories, pinning important articles,
knowledge_admin	Knowledge administrators can create new knowledge bases and manage all KB
admin	Service now Administrators can configure knowledge workflows, set knowledge properties, and manage knowledge forms and homepages.
KM Coach	Review Link Quality, Performs AQI Surveys, Leverages AQI Coach knowledge team

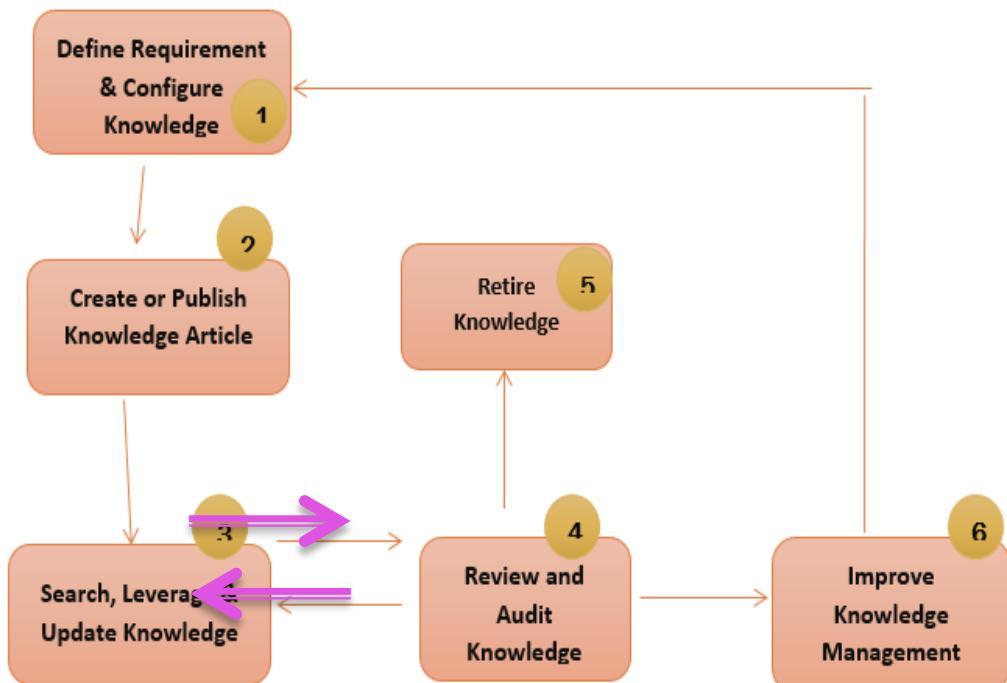
Knowledge Management Usage and Benefits

- This application will save time and money for user or customers
- Reducing the time duration employees spend searching for documents and information,
- It will minimize resolution times, and empowers knowledge users and consumers to increase incident deflection.
- Reducing repeated kind of services following by knowledge articles
- Self-service perspective, through knowledge articles, communications, the ability to interact with other users, and through automated-support.
- Find the required answer fast followed by KA
- Articles are help more to Help Desk agents and Customers to full fill self-service opportunities
- Increase user satisfaction
- Promote Knowledge Sharing
- Drive Continuous Improvement
- Track usage and governance

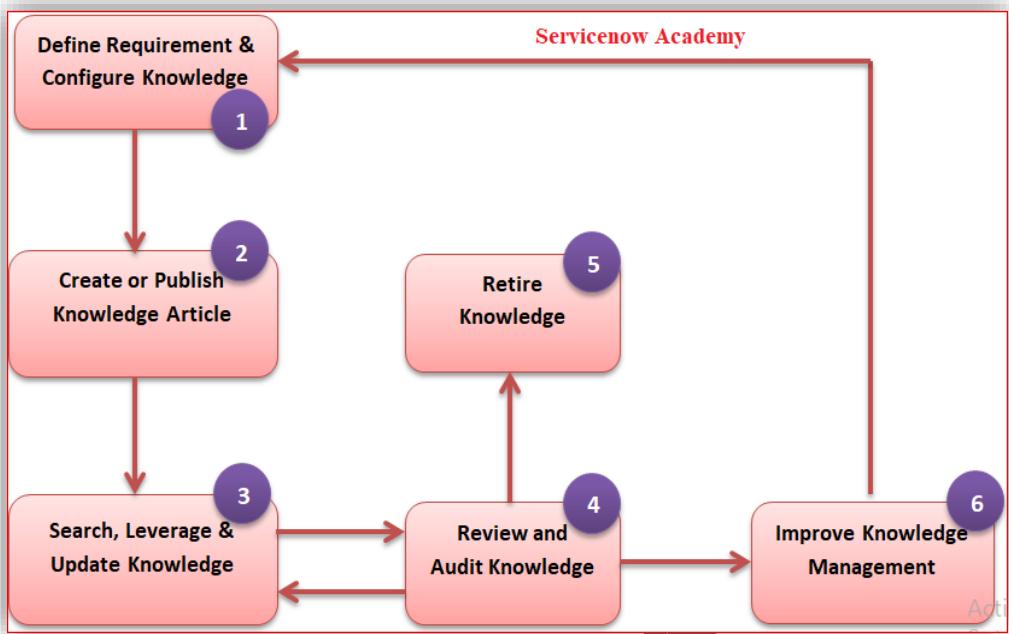
KM Architecture

Knowledge management application going to follow certain procedure to **Create, Publish and Retired** of desired articles

Follow KM process flow in below diagram

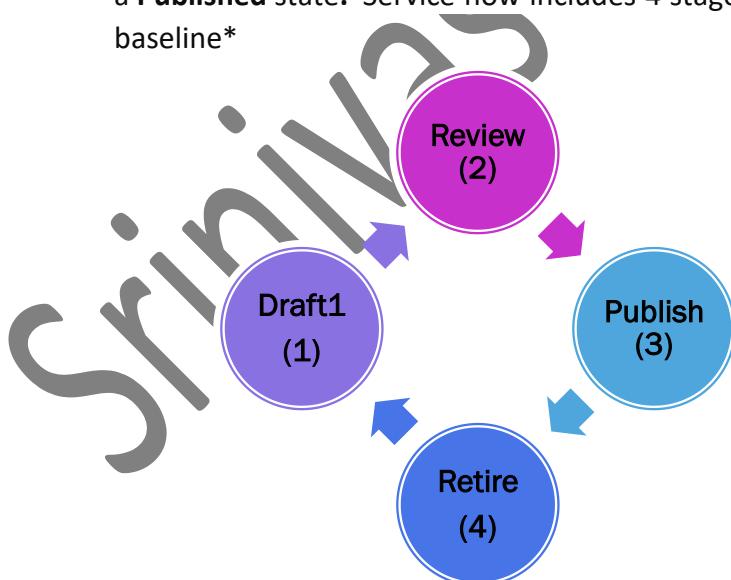


Srinivas



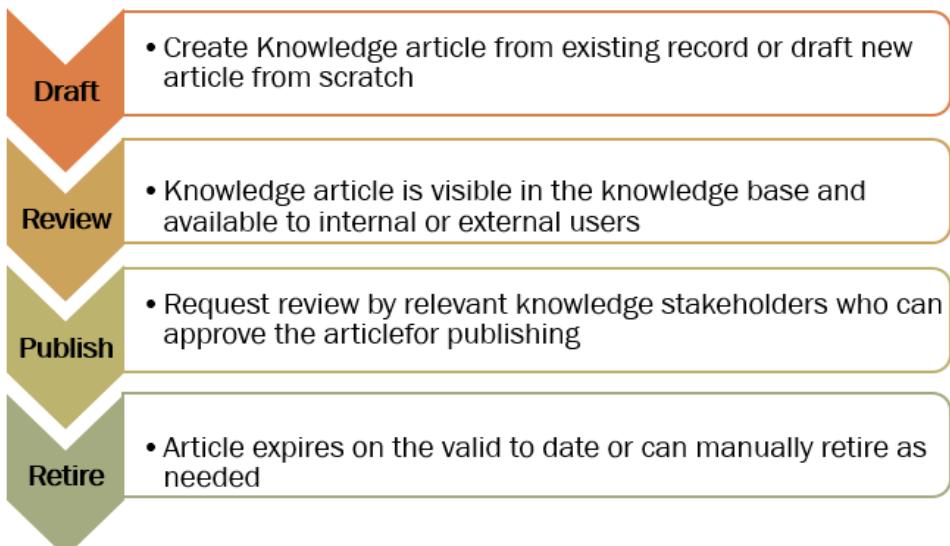
Knowledge Management Life Cycle

Knowledge management consist different states to move our knowledge article. By default, to be able to search for and find a knowledge article, it must be in a **Published** state. Service now includes 4 stages for a knowledge article in its baseline*



KM Workflow Stages	
Draft	Knowledge base contributors create draft knowledge articles independently (or from existing task records with the Knowledge Advanced plugin*)
Review	Subject Matter Experts (SMEs) and/or knowledge managers review the knowledge article for content accuracy and formatting consistency.
Publish	Once the knowledge article is reviewed and approved, the article is published to the knowledge base.
Retire	The knowledge manager may retire the article once it reaches its valid-to date, or re-engage with SMEs to publish a new version.

Knowledge Management State Model



Scope and Requirements

To get started with Knowledge Management, your organization must define the knowledge scope and requirements. Consider the following:

1. What are the objectives of your organization? Think about both quick wins and long term goals.
2. Identify your knowledge consumer groups. These groups will define the requirements for the **User Criteria** records in Service Now. Examples include:



- External Customers
- Internal End Users
- Service Desk Members
- HR Employees
- Users within a specific region



3. How should you structure your knowledge base(s)? Every organization's needs are different. Understanding ServiceNow's knowledge **access** capabilities will help you make that decision. Knowledge visibility can be restricted to the knowledge base level, the knowledge article level, and even at the field level within a knowledge article. Once the knowledge bases are defined, you will need to determine:



1. The **categorization** schema
2. The knowledge base author(s), owner and managers
3. The publish and retire workflows for the knowledge bases
4. When can knowledge be created?
5. What notifications are needed and who should receive them?
6. Will consumers be able to provide feedback? How will feedback be managed?
7. Understand the risks within your organization. What areas have the weakest documentation and rely the most on specific individuals for work to be done?
8. Will you use knowledge **templates**? Are you using them today? What are the key attributes of your knowledge? These attributes should be defined in your templates.
9. Where is your organization's knowledge sitting today? You will need to determine if you should **migrate** current articles to service now, create **new** knowledge articles, or if there should be integration to service now with your knowledge repository.



1. As part of this exercise, evaluate the accuracy and validity of current knowledge to determine what should stay, what should go, and what needs to be updated.

Continue reviewing this course to get an overview of Knowledge Management in service now.

What is Knowledge Base?

1. A knowledge base acts as a container of knowledge articles in Service Now.
2. Knowledge bases are the highest level of organization and access control for knowledge articles.
3. Each knowledge base can have its own **owner**, **manager**, **publish workflow**, retire workflow
4. Knowledge base can have **commenting**, **suggesting**, **category editing**, and social question and answering **active or inactive**.

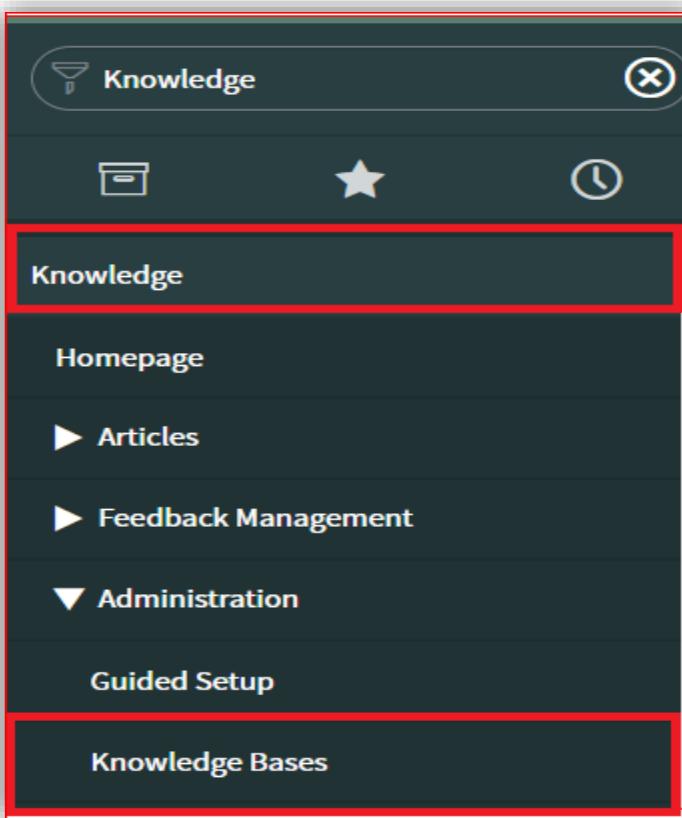
Exercise 1: Create Knowledge Base

Let's start create a new knowledge base in service now

Procedure

1. Navigate to **Knowledge** → **Administration** → **Knowledge Bases**

Srinivas



2. Click on New Button

A screenshot of the "Knowledge Bases" list view. The top navigation bar includes a "New" button, which is highlighted with a red box. There are also "Search" and "Order" buttons. The main area shows a table of knowledge bases. A red box highlights the first row of the table. The columns are: Title, Description, and Owner. The rows show the following data:

	Title	Description	Owner
<input type="checkbox"/>	Social QA	Default knowledge base for Service Porta...	System Administrator
<input type="checkbox"/>	IT	The ACME North America IT Service Desk K...	Bernard Laboy
<input type="checkbox"/>	Knowledge	All existing knowledge articles prior to...	System Administrator

3. Fill Knowledge Base Form

Srinivas Suryanarayana

The screenshot shows the 'Knowledge Base' configuration page. Key fields include:

- Title:** HR Knowledge Base (highlighted with a red box)
- Application:** Global
- Owner:** Fred Luddy
- Managers:** David Loo
- Publish workflow:** Knowledge - Instant Publish (highlighted with a red box)
- Retire workflow:** Knowledge - Instant Retire (highlighted with a red box)
- Active:** Checked
- Description:** This knowledge base will maintain all HR related knowledge articles

4. Provide unique name of the knowledge base (**HR Knowledge Base**)
5. Publish Workflow: **Knowledge-Instant Publish**
6. Retire Workflow: **Knowledge-Instant Retire**
7. Click on **Submit**

Knowledge Bases			
	New	Search	Order
<input type="checkbox"/> All > Active = true			
<input type="checkbox"/>	<input type="checkbox"/> Title	<input type="checkbox"/> Description	<input type="checkbox"/> Owner
	Search	Search	Search
<input type="checkbox"/>	<input type="info"/> HR Knowledge Base	This knowledge base will maintain all HR ...	<u>Fred Luddy</u>
<input type="checkbox"/>	<input type="info"/> Social QA	Default knowledge base for Service Porta...	<u>System Administrator</u>
<input type="checkbox"/>	<input type="info"/> IT	The ACME North America IT Service Desk K...	<u>Bernard Laboy</u>
<input type="checkbox"/>	<input type="info"/> Knowledge	All existing knowledge articles prior to...	<u>System Administrator</u>

8. New knowledge Base was created successfully

Note: Once the knowledge base is created and saved, additional configurations can be set for the knowledge base.

Exercise 2: Create Knowledge BaseCategory

List of knowledge categories and sub-categories associated with this knowledge base.

Procedure

1. Navigate to **HR Knowledge Base** → **Knowledge Categories**
2. Click on **New**

The screenshot shows a navigation bar with tabs: Knowledge, Questions, Can Read, Can Contribute, Featured Content, and Knowledge Categories. The 'Knowledge Categories' tab is highlighted with a red box. Below the navigation bar is a search bar with fields for 'Search' and 'Label'. A green 'New' button is also highlighted with a red box. The main content area displays a message 'No records to display'.

3. Label: Payroll
4. Value: Payroll
5. Click on **Submit**

The screenshot shows a 'Knowledge Category' form titled 'New record'. It has fields for 'Label' (containing 'Payroll'), 'Value' (containing 'Payroll'), 'Parent ID' (containing 'Knowledge Base: HR Kn' with a search icon), and 'Active' (with a checked checkbox). At the bottom is a 'Submit' button, which is highlighted with a red box.

Knowledge Category
New record

* Label	<input type="text" value="Payroll"/>
Value	<input type="text" value="Patroll"/>
Parent ID	Knowledge Base: HR Kn <input type="button" value="Search"/> <input type="button" value="Info"/>
Active	<input checked="" type="checkbox"/>
<input type="button" value="Submit"/>	

6. Create one more Knowledge Base Category(On-Boarding)

Knowledge Category
New record

* Label	<input type="text" value="On-Boarding"/>
Value	<input type="text" value="On-Boarding"/>
Parent ID	Knowledge Base: HR Kn <input type="button" value="Search"/> <input type="button" value="Info"/>
Active	<input checked="" type="checkbox"/>
<input type="button" value="Submit"/>	

The screenshot shows the 'Knowledge Categories' section of a software interface. At the top, there are tabs: Knowledge, Questions, Can Read, Can Contribute, Featured Content, and Knowledge Categories (2). The 'Knowledge Categories' tab is highlighted with a red border. Below the tabs is a search bar with a dropdown menu set to 'Label'. A green 'New' button is visible. The main area displays two categories: 'On-Boarding' and 'Payroll', each with a small icon and a disclosure arrow.

Exercise 3: Create New Knowledge Article

The list of knowledge articles stored in this knowledge base. New knowledge articles can be created from knowledge tab

Procedure

1. Navigate to **HR Knowledge Base** → **Knowledge**
2. Click on **New**

The screenshot shows the 'Knowledge' section of a software interface. The 'Knowledge' tab is highlighted with a red border. Below the tabs is a search bar with a dropdown menu set to 'Number'. A green 'New' button is visible. The main area displays a message: 'Knowledge base = HR Knowledge Base'. Below the message are several filter options: 'Number', 'Short description', 'Author', 'Category', and 'Workflow'. At the bottom, it says 'No records to display'.

3. Fill **Knowledge Article Form**

Number	KB0010003	Article type	Wiki
* Knowledge base	HR Knowledge Base	Workflow	Draft
Category	On-Boarding	Source Task	
Valid to	2022-02-18	Attachment link	<input checked="" type="checkbox"/>
Display attachments <input checked="" type="checkbox"/>			
* Short description	Employee On-Boarding		

4. Knowledge base: **HR Knowledge Base**
5. Category: **On-Boarding**
6. Valid to :**2022-02-18**
7. Article type: **Wiki**
8. Wikitext: **Post some content into your article**

Wiki:  Wikitext

- +

What is onboarding? We've all experienced onboarding when starting a new job. You head to the office on your first day ready to get started. You'll probably get right to that cool project you discussed with your new manager during the interview, right? Not so fast.

Welcome a new hire using our onboarding checklist

When you arrive, you're quickly handed over to an HR representative for onboarding. They probably give you a tour of the office and introduce you to everyone before taking you into a room to complete common onboarding tasks like:

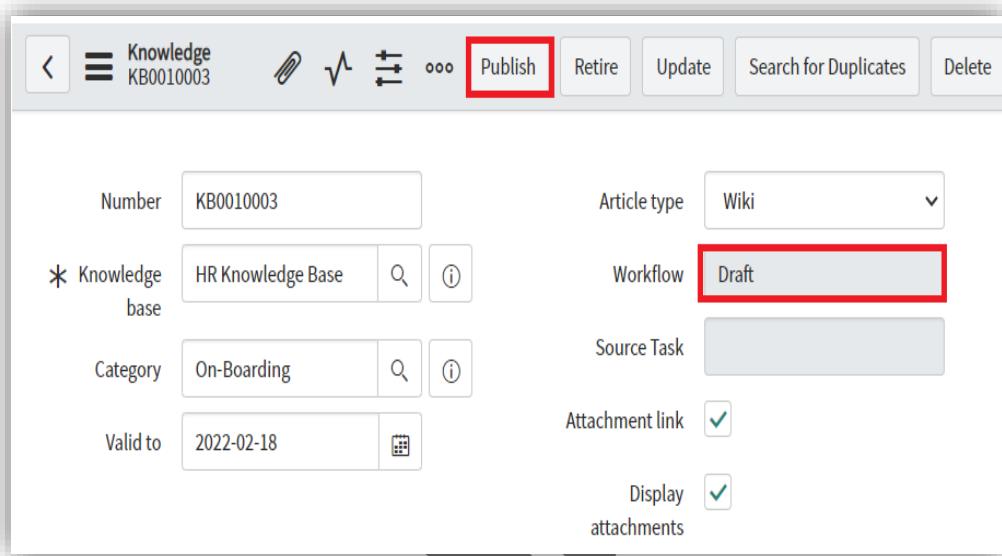
9. Click on **Submit**
10. Publish your Knowledge Article
11. Click on **Publish**

Exercise 4: Publish Knowledge Article

When we created a knowledge article the by default state is **Draft**

Procedure

1. Click on **Publish** button to publish your article (with out approval)
2. Publish workflow got select (**Instant Publish**) for this **knowledge base**



3. Check your knowledge article in **Incident Table**
4. Navigate to Incident → Click on Create New Module
5. Type **Employee On-Boarding** in short description filed
6. Find our knowledge article here

The screenshot shows the 'Incident' module. On the left, there is a sidebar with filters: 'Assigned to me', 'Open', 'Open - Unassigned', 'Resolved', 'Closed', 'All', and a refresh icon. The main area has a search bar with 'Employee On-Boarding' and a 'Related Search Results' dropdown. A knowledge article result is displayed: 'Employee On-Boarding' (Short description), 'Employee On-Boarding' (Description), and a detailed preview: 'Learning company policies covered in the employee handbook Learning the organizational chart and common company procedures These are only a few high...'. The preview also shows the 'HR Knowledge Base' source, author 'System Administrator', views (1), last modified (2020-06-06), and rating (4 stars).

Exercise 5: Retire Knowledge Article

We can retire the latest published knowledge article. Retiring a knowledge article does not create a new version. It simply marks the article as Retired.

Procedure

1. Navigate to **HR Knowledge base** → **On-Boarding** → **Employee On-Borating**
2. Knowledge article will be retire depends **on Valid to or**
3. Retire your knowledge article forcefully
4. Click on **Retire** Button
5. Article will be retired (**without approval**) Instant Retired

The screenshot shows the 'Knowledge' edit screen for article KB0010003. The 'Retire' button in the top right toolbar is highlighted with a red box. The form contains fields for Number (KB0010003), Article type (Wiki), Knowledge base (HR Knowledge Base), Category (On-Boarding), Published (2020-06-06), Valid to (2022-02-18), Workflow (Published), Source Task, Attachment link (checked), and Display attachments (checked). The 'Valid to' field is also highlighted with a red box.

Note: Once Knowledge article is retired, the article is not visible any customers until to republish it

The screenshot shows the 'Knowledge' edit screen for article KB0010003 after retirement. The 'Republish' button in the top right toolbar is highlighted with a red box. The form fields are identical to the previous screenshot but reflect the 'Retired' status: Workflow is set to 'Retired' and the 'Republish' button is now active.

User Criteria

We specify user criteria for a knowledge base to restrict which users are granted access to read and contribute knowledge articles to that knowledge base.

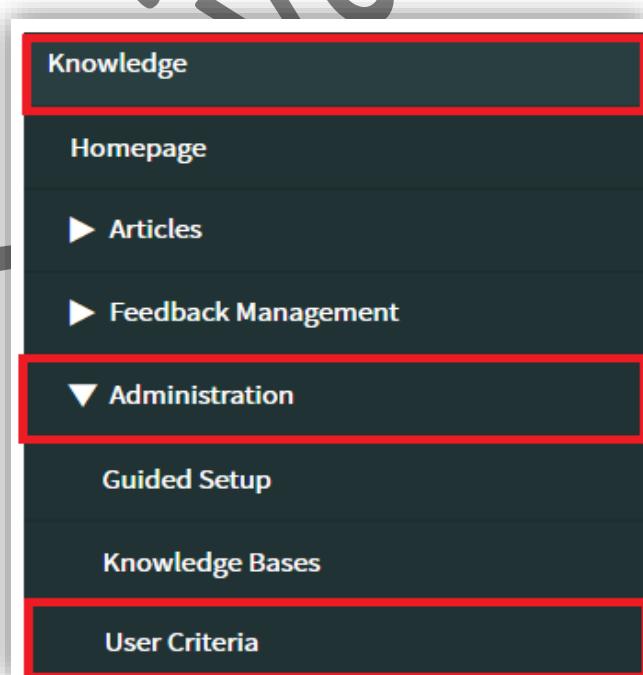
You can assign user criteria to control **read** or **contribute** access. For example, you could enable all users in your Support department to have contribute access to a knowledge base by creating a user criteria record with the required department set to Support, and then including the user criteria record in the Can Contribute user criteria.

Exercise 6: Configure User Criteria for Our Knowledge Base

Let's start create a new user criterion for our knowledge base
Visible to only **HR Department** people and company is **ACME China**

Procedure

1. Navigate to **Knowledge**→**Administration**→**User Criteria**
2. Click on **New Criteria Record**



3. Fill user criteria form

User Criteria may be used to restrict access to records in Service Catalog and Knowledge

* Name **HR Knowledge User Criteria**

Application Global

Active

Users

Companies ACME China

Groups

Locations

Roles

Departments HR

Advanced

Match All

Submit

4. Name: **HR Knowledge User Criteria**
5. Companies: **ACME China**
6. Department:**HR**
7. Click on **Submit**
8. Add your new user criteria to **HR Knowledge Base**
9. Open **HR Knowledge Base Record**
10. Open **Can Read Tab**
11. Click on **Edit** button
12. Add your **Criteria**

Knowledge	Questions	Can Read	Can Contribute	Featured Content	Knowledge Categories
Knowledge (1)	Questions (1)	Can Read (1)	Can Contribute (1)	Featured Content	Knowledge Categories (2)
Can Read New Edit... Search for text ▾ Search					
1 to 1 of 1					
Knowledge Base = HR Knowledge Base					
Can Read					
HR Knowledge User Criteria					

13. Check your user criteria whether it is working properly or not

Related List	Description
Can Read	Users can read knowledge articles in the knowledge base.
Cannot Read	Users can't read knowledge articles in the knowledge base.
Can Contribute	Users can create, modify, and retire knowledge articles in a knowledge base. Contribute access to a knowledge base also provides read access to all articles in the knowledge base.
Cannot Contribute	Users can't create, modify, retire, or read knowledge articles in the knowledge base

1. In the selected related list, add the required user criteria.
2. To add a new user criteria record, **click New**, specify the required fields, and then **click Submit**.
3. To add an existing user criteria record, **click Edit**, move the required user criteria from the Collection column to the Knowledge column, and then **click Save**.
4. On the Knowledge Base form, **click Update**.

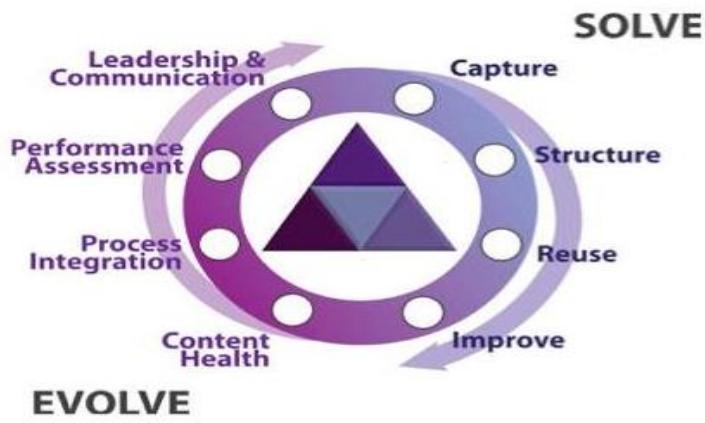
Knowledge –Centered Service (KCS)

Knowledge-Centred Service configuration (KCS) is a method for the creation and continuous improvement of knowledge based on the experience of agents and the patterns observed by knowledge reuse.

1. Knowledge Management provides features
2. To improve knowledge article quality reviews,
3. In-context knowledge capture
4. Feedback management.
5. Article Quality Index
6. Knowledge demand insights

KCS strongly supports and enables the **Solve & Evolve Loop** through reward learning (a positive reinforcement methodology), collaboration, sharing and improving knowledge by enabling both agents and end users to interact with knowledge by flagging, fixing, updating, and creating new knowledge.

KCS Double Loop Process



The Solve Loop includes resolution activities that work to capture and re-use resolution steps. The Evolve Loop focuses on assessment and improvement activities. Together the Solve & Evolve Loops create a self-correcting system that is co-dependent of one another.

Throughout this course we will discuss ways to implement KCS principles in Service Now. While not every organization will be mature enough in their Knowledge Management journey to implement all parts of KCS right away, there will be pieces that can and should be leveraged at any point of maturity.

Leveraging Knowledge Management in Service Now will:

- Reduce documentation search time
- Increase incident and case deflection
- Reduce resolution and fulfillment time for incidents and cases
- Enhance resolution collaboration across teams
- Improve customer productivity and overall user experience
- Improve resolution quality and consistency
- Support the activities within the Solve & Evolve Loops

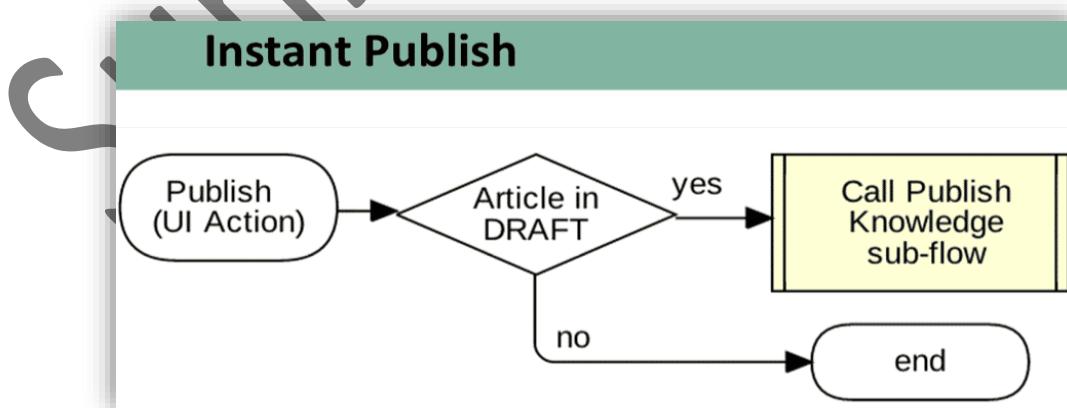
Knowledge Management Workflows

ServiceNow follows workflow automation to publish and retire knowledge articles. There are two defined workflows available as part of the baseline for both publishing and retiring a knowledge article. These baseline workflows can be used as-is or customized for your environment.

Publishing a knowledge article

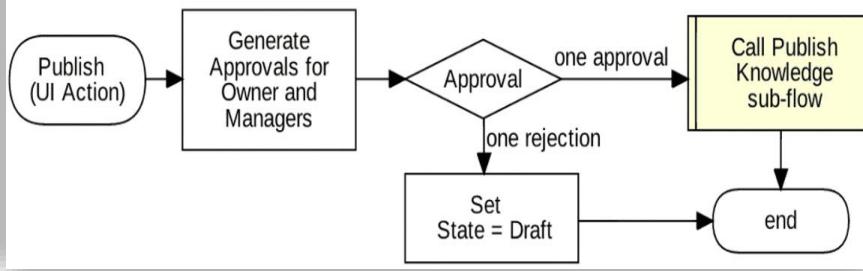
The Publish workflow will move a knowledge article between the **Draft** and **Publish** stages. It defines what, if any, approvals are required to move stages. It will also define if any notifications are required, and who they are sent to.

By default, there are two publish workflows available for knowledge:



Note: Publishes articles in the knowledge base without **requiring** an approval.

Approval Publish

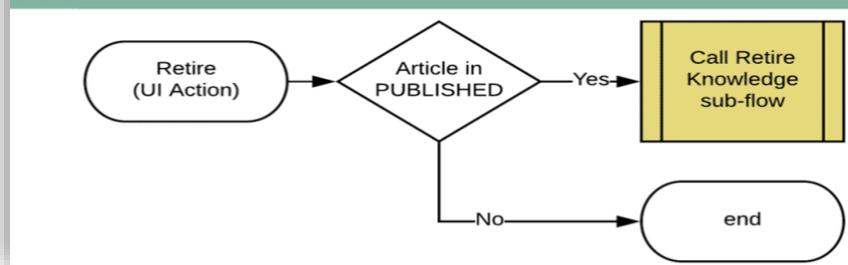


Retiring a knowledge article

The Retire workflow will move a knowledge article between the Publish and Retire stages. It defines what, if any, approvals are required to move stages. It will also define if any notifications are required, and who they are sent to.

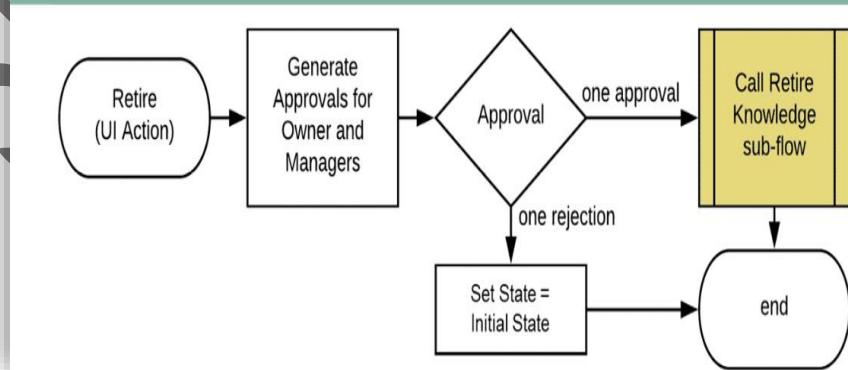
By default, there are two retire workflows available for knowledge:

Instant Retire



Note: Retires articles in the knowledge base without **requiring an approval**.

Approval Retire



Note: Requests approval from the manager(s) and owner of the knowledge base before moving the articles to the retired state.

Srinivas Sunkara
Thank You