

12-2017

Customer-Centric Service Management Using Servicenow

Veenadhari Kotha

St. Cloud State University, vkotha@stcloudstate.edu

Follow this and additional works at: https://repository.stcloudstate.edu/msia_etds

Recommended Citation

Kotha, Veenadhari, "Customer-Centric Service Management Using Servicenow" (2017). *Culminating Projects in Information Assurance*. 35.

https://repository.stcloudstate.edu/msia_etds/35

This Starred Paper is brought to you for free and open access by the Department of Information Systems at theRepository at St. Cloud State. It has been accepted for inclusion in Culminating Projects in Information Assurance by an authorized administrator of theRepository at St. Cloud State. For more information, please contact rswexelbaum@stcloudstate.edu.

Customer-Centric Service Management Using Servicenow

by

Veenadhari Kotha

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

in Information Assurance

December, 2017

Starred Paper Committee:
Dennis Guster, Chairperson
Susantha Herath
Sneh Kalia

Abstract

ServiceNow enterprise IT service automation platform combines ITIL v3 process support, platform-as-a-service (PaaS) delivery, and Web functionality, to provide a flexible, intuitive and self-managing application (Muir, 2014). Fundamentally, it is a service management ticketing tool for every department in the enterprise that offers functional flexibility, relative ease of deployment, and competitive pricing (About ServiceNow: TechWeb, n.d.). ServiceNow is a PaaS model, where a platform is also supplied, allowing customers to develop and extend the OOB (Out of the box) functionality. With unique pre-built services and reusable components, we can rapidly build, integrate, and extend the features or scope of our applications to meet the dynamic requirements of our business. The goal of this paper is to construct an optimal and efficient solution by utilizing the ServiceNow provided functionality and customize it to meet the business needs.

Acknowledgements

First, I wish to thank the members of my academic committee, Dr. Dennis Guster, Dr. Susantha Herath, and Dr. Sneh Kalia, whose valuable feedback and advice proved to be a milestone in the accomplishment of this paper.

I would also like to pay warmth and gratitude to my family and friends, without whose support, this paper would not have been completed.

Table of Contents

	Page
List of Tables	6
List of Figures	7
Chapter	
I. Introduction	14
Introduction	14
Problem Statement	15
Nature and Significance of the Problem	15
Objective of the Study	16
Study Questions	16
Definition of Terms	16
Summary	17
II. Background and Review of Literature	18
Introduction	18
Background Related to the Problem	18
What is a Production Request?	18
What is Dynamics SL?	19
What is JIRA?	19
Creating a Job in Dynamics SL	19
Production WOR	28
WO	29

Chapter	Page
Literature Related to Problem	33
Literature Related to the Methodology	34
Summary	38
III. Methodology	39
Introduction	39
Design of the Study	39
Data Collection	39
Data Analysis	39
IV. Data Presentation and Analysis	42
Introduction	42
Data Presentation	42
Data Analysis	62
Summary	131
V. Results, Conclusion, and Recommendations	132
Introduction	132
Results	132
Conclusion	133
Future Work	134
References	135
Appendix	137

List of Tables

Table	Page
1. Definition of Terms Used in This Paper	16
2. Job Types, Codes and Description	22
3. JIRA Components and Description	24

List of Figures

Figure	Page
1. Windows menu	20
2. JIRA link menu	21
3. Create new project	23
4. Create issue	23
5. Select component	24
6. Select NHBC	24
7. Specify job summary	25
8. Job creating page	26
9. Watchers	27
10. Adding watchers	27
11. Back to issue	27
12. Work order (WO) general info	28
13. Specifying dates	28
14. Specifying NHBC	29
15. Watchers added	29
16. Relativity export	30
17. Business rule—when to run	43
18. Business rule—actions	44
19. Business rule—advanced	45
20. Script include	46

Chapter	Page
21. Script include–structure	48
22. Script include–call	48
23. Script action	48
24. Client script	51
25. UI script	52
26. Catalog client script	53
27. UI action	54
28. UI action	56
29. UI policy–when to apply	57
30. UI policy–script	58
31. UI policy–UI policy actions	59
32. UI policy action	59
33. Sample graphical workflow editor	61
34. Database design for SD job	63
35. Workflow design for new matter	64
36. New matter request workflow	65
37. Workflow activity–set values	67
38. Workflow activity–if	67
39. Workflow activity–create task–basics	68
40. Workflow activity–create task–populate variables	69
41. Workflow activity–create task–schedule	69

Figure	Page
42. Workflow activity–create task–script	70
43. Workflow activity–join	71
44. Workflow design for job initiation	72
45. Job initiation workflow	73
46. Intake request workflow	74
47. Intake request workflow	75
48. Intake request workflow–generate evidence	75
49. Intake request cancellation workflow	76
50. Image sweep request workflow	77
51. Client archive request workflow	78
52. Client archive request workflow	79
53. Set status to complete if inactive–when to run	80
54. Set status to complete if inactive–actions	80
55. Set status to complete if inactive–advanced	81
56. System property–ihe.job.include	82
57. Generate FQ job code	83
58. Generate FQ job code	84
59. Set Client	84
60. Deactivate metric	85
61. Calculate SD metrics from job	85
62. Update linked jobs	86

	10
Figure	Page
63. Update linked jobs	87
64. Evidence update	88
65. Create entire corpus field list	89
66. EC fields list	90
67. EC fields list	91
68. EC fields list	92
69. Set FPNI query on PR	92
70. Image sweep autofill fields	93
71. Sync IS task fields	93
72. Generate client facing volume	94
73. Check validity before submit	94
74. Check conditions before submitting	95
75. Set order on field cond builder	95
76. Set lode file field labels	96
77. Capture query for exception	96
78. AIM_getMetricBilling	97
79. AIM_getMetricBilling	98
80. AIM_getMetricBilling	98
81. AIM_getMetricBilling	99
82. AIM_getMetricBilling	100
83. AIM_getMetricBilling	100

Figure	Page
84. AIM_groupQuery	101
85. AIM_GetLocaiton	101
86. AIM_unique_FQJOBcode	102
87. AIM_CascadeCommentsToParent	102
88. AIM_MandatoryFields	103
89. IheConditionHolder	104
90. IheConditionHolder	104
91. IheConditionHolder	105
92. IheConditionHolder	105
93. Check unique FQ job code	106
94. Hide annotations if asset	107
95. Hide annotations if asset	108
96. Autofill suffix on PR	108
97. Show/hide endorsements	109
98. Choices based on sweep type	109
99. Autofill dept. & site reported	110
100. Resolution type options	110
101. COC & calculate size of GB	111
102. Check attachments	111
103. Validate matter code	112
104. Default choices based on type	113

Figure	Page
105. Default choices based on type	114
106. Show/hide load file related list	114
107. Cancel job	115
108. Spec ready	116
109. Spec ready	117
110. Complete check in	117
111. Back to draft	118
112. Create metrics	119
113. Create metrics	120
114. Create metrics	121
115. Don't bill	121
116. Billing override	122
117. Create intake request	123
118. Cancel QINC	124
119. Escalate	124
120. In progress	125
121. Unassign	125
122. Input needed	126
123. Input provided	127
124. Complete task	128
125. QC pass	128

Figure	Page
126. QC fail	129
127. Create quality incident	129
128. Copy components to media type	130

Chapter I: Introduction

Introduction

Anyone who's involved in the IT industry will have heard of ServiceNow. It is a cloud-computing company, founded in 2003 by Fred Luddy, that offers platform-as-a-service (PaaS) enterprise service management software for various departments within an organization including but not limited to human resources, law, facilities management, sales and marketing, and even IT (ServiceNow—What is It and what are the latest trends, 2014). ServiceNow is an internationally acclaimed company with its presence in North America, Latin America, Europe, The Middle East and Africa, Asia-Pacific, and Japan. Though it has its roots in IT, we can customize ServiceNow to come up with efficient solutions for any domain in the company. It is a system where users across the company come together to get the work done.

It is this very aspect of ServiceNow that will be applied in this paper to develop a solution for an e-Discovery project that identifies, collects and produces electronically stored information (ESI) for the purpose of civil litigation or an investigation (The basics: What is e-Discovery?, n.d.). Depending on the type of request, different tools are used to extract the data, but the process leading to the decision about the appropriate tool comprises of word documents, excel sheets, emails, paper documents and such. It is inefficient, time-consuming, insecure since everyone has access to the files and highly prone to error since it is all managed manually. This paper focuses on employing ServiceNow to replace that process thereby ensuring data integrity, confidentiality and in a way availability as well.

Problem Statement

Customers now want solutions that are available faster, and accessible anywhere. This company spends about 40-50 hours per Product Manager team per month gathering specifications and handling them (excluding the time spent on the actual e-Discovery), for the e-Discovery for various legal matters. This paper focuses on How ServiceNow, a cloud-based ITSM technology, can be customized to provide the client a platform to manage their specifications more efficiently.

Nature and Significance of the Problem

The company has dedicated tools to perform the actual act of e-Discovery depending on the type of job undertaken. Hence, deciding the type of job is highly essential. So far, the company has been spending a tremendous amount of staff-hours into meeting with the clients, collecting the specifications of the job, generating unique identifiers for each job type and each job, document handling, meeting the client again if there are any changes and final check. The tickets are tracked through JIRA, and the unique identifiers are generated in Dynamic SL. The price of e-Discovery for every job is calculated based on the no. of hours spent on the actual discovery. So, spending massive amounts of time to get to that point implies that the productivity is low. If there's a solution to reduce this time and provide an efficient way, then it means the company can take on more jobs and thereby the profit marginally increases.

Though here the problem is explained concerning the current project, we can apply it to any project in an organization, not just within IT. Service automation is a

revolution—ServiceNow can be used to provide service automation for any department.

Objective of the Study

The purpose of this study is to implement ServiceNow to develop an efficient, cost-reducing solution that helps in the process of e-Discovery, providing breathing space for improvement and new features.

Study Questions

- How does ServiceNow work?
- How can the current system be migrated to ServiceNow without loss of data?
- How can ServiceNow be used to replace JIRA, Dynamics SL and all the other media currently used?
- How can ServiceNow be used to achieve data confidentiality, integrity, and availability?

Definition of Terms

Table 1

Definition of Terms Used in This Paper

Client	The company that requests for e-Discovery of their ESI
Matter	The civil litigation or the legal case for which e-Discovery is being done
Job	Each e-Discovery task performed for a Matter
ESI	Electronically Stored Information
IT	Information Technology
ITIL	IT Infrastructure Library
SSO	Single Sign-On
HR	Human Resources
DB	Data Base
OOB	Out of the Box
BR	Business Rule
CS	Client Script
RFC	Request For Change
BR	Business Rule

Summary

In this chapter, we briefly introduced what ServiceNow is, the objective of this paper, the problem with the existing process and different project-specific terms. In the next chapter, we will learn more about ServiceNow and a review of the literature, which includes, an elaborate discussion of ServiceNow capabilities and documentation, i.e., the proposed system and an in-depth explanation of the current implementation, i.e., the problem.

Chapter II: Background and Review of Literature

Introduction

In this chapter, we are going to learn in detail how the e-Discovery process is currently implemented in the project by studying an example, i.e., go through the process of e-Discovery starting from specification gathering to the billing stage and learn the literature and working of the tools in the present system. We will also learn about the working of ServiceNow, what it does, and review the related literature.

Background Related to the Problem

Let us first understand the basic structure of the project. A client is the end company that requests for e-Discovery for the purpose of a legal case. A Matter is a legal case or civil litigation in question. A Job is a single instance of e-Discovery process/request. Each client can be involved with multiple Matters. And in each Matter, the Client may request for multiple Jobs. There are also various types of jobs including Production Request, Processing Request, Custom Request, Intake Request, etc. Of these, Production Request is the most important and elaborate one.

What is a Production Request?

A Production Request served by one party to an action on another (as under Federal Rule of Civil Procedure 34) for the presentation for inspection of specified documents or tangible things or for permission to enter upon and inspect land or property in the other party's possession (Request for Production–FindLaw, n.d.). Any data that is stored in an electronic form may be subject to production under common e-Discovery rules.

What is Dynamics SL?

Microsoft Dynamics SL is one of Microsoft's ERP (Enterprise Resource Planning) software products designed to meet the specific needs of project-driven, service-driven and distribution driven businesses. It provides ERP solutions for various industries including, but not limited to Computer and IT related services, Government Contracting, Management consulting, etc. (Microsoft dynamics SL, n.d.).

What is JIRA?

JIRA is an issue tracking, bug tracking, project management tool that utilizes the software-as-a-service model. The JIRA dashboard consists of various useful functions and features which make handling of issues easy (JIRA software overview, n.d.).

Creating a Job in Dynamics SL

Overview. All client requests should be tracked in JIRA for billing purposes. Except for asset requests, HS tickets, and Prod Ops requests which can be submitted easily through email aliases, standard review volumes, productions, imaging jobs, etc., all else need to be created through Microsoft Dynamics SL.

Procedure.

1. Open Microsoft Dynamics SL.

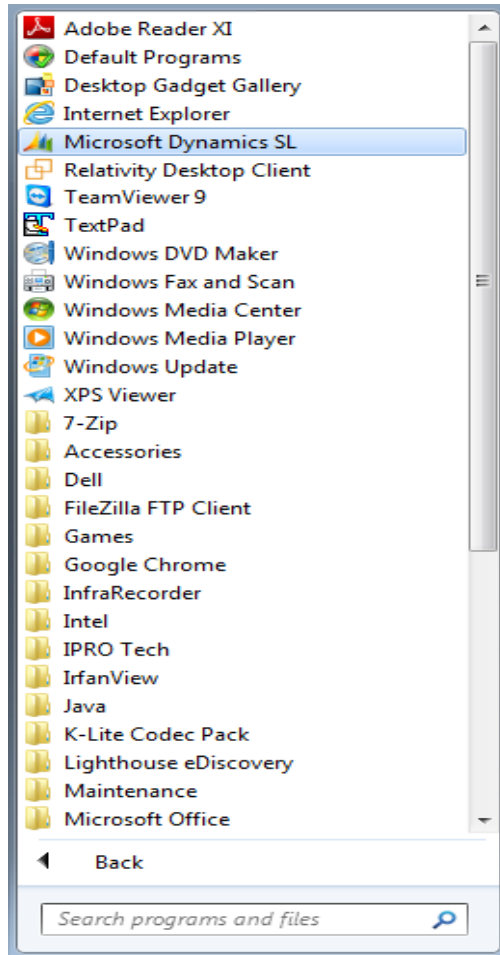


Figure 1. Windows menu.

2. Navigate to JIRA Integration Group and click on "JIRA Integration Link <--
CLICK HERE."

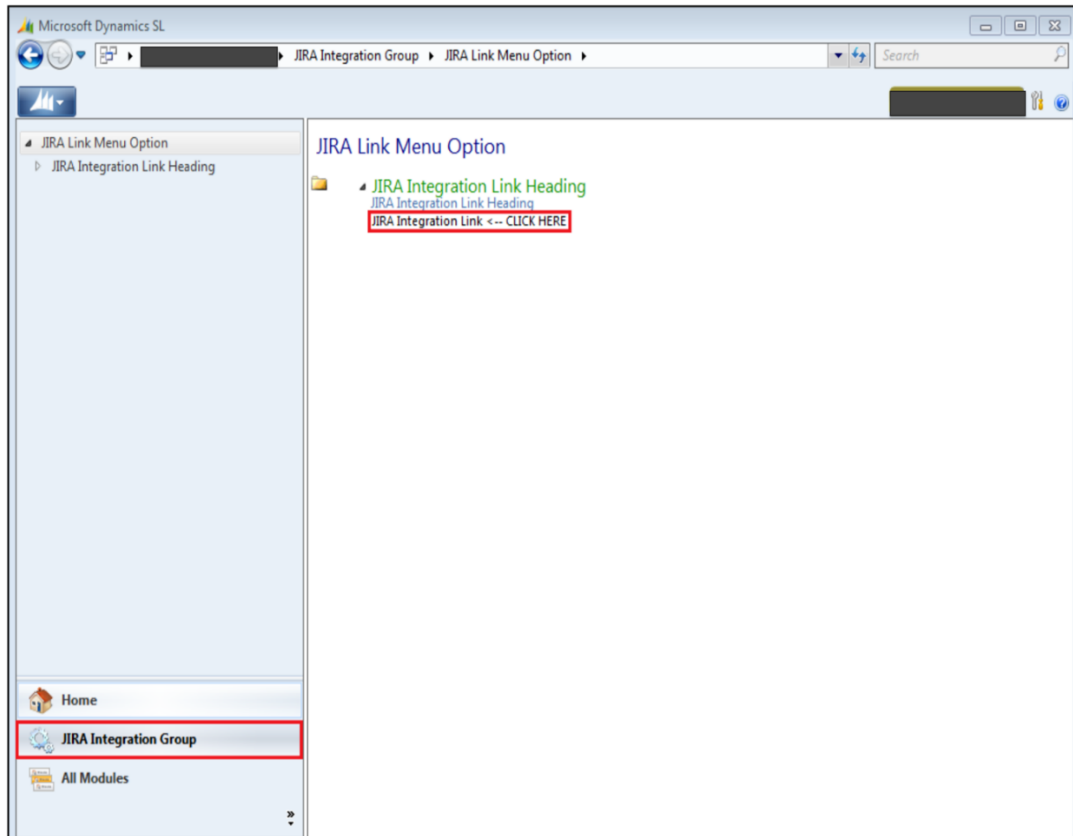


Figure 2. JIRA link menu.

3. A “**Dynamics SL ↔ JIRA**” page will open in the browser
4. Use the drop-down menu to select the appropriate **Customer ID** (Client)
5. Use the drop-down menu to select the appropriate **Master Project ID** (Matter/Case)
6. **Client Matter ID** will auto populate with the Matter number if one exists
7. Enter the appropriate **Client Facing Volume**
8. Use the drop-down menu to select the appropriate **Job Type Code**

Table 2

Job Types, Codes and Description

Job Type Code	Examples
AR - Client Archive Request	Requests to export a Relativity database or subset for archiving purposes
CR - Custom Request	Requests that fall outside of the company's standard workflow and require a custom solution (use conservatively)
DC - Decryption	Requests to process decrypted natives such as loose emails (verify with senior resource before utilizing)
FD - Focus Discovery	Requests to track non-hourly billable components for matters involving the Focus Discovery team
HC - Scan Documents	Requests to scan documents by the third party vendor Sound Legal
HS - Hosted Solutions	Requests relating to Relativity or a SmartSeries component that are not part of an existing ticket
IM - Imaging	Requests for image documents for loading into Relativity
IS - Image Sweep Request	Requests for setting up the company's auto-imaging sweeps in Relativity and eCapture (AI, RT, PT)
ME - Major Exceptions	Requests for repair and process major exceptions encountered during an RV request
NR - Native Redaction	Requests to image natively redacted documents (unique to specific matters only - verify with senior resource before utilizing)
PP - Pre-Processed Data	Requests to load pre-processed data into Relativity (the third party normalize and upload workflow only)
PR - Production Request	Requests to produce documents
PS - Professional Services	Requests involving a TPM or Solutions Analyst that would not hit Production, Processing, or HS
RE - Reuse	Requests to propagate images to duplicate records in Relativity (unique to specific matters only - verify with senior resource before utilizing)
RV - ESI Review Volume	Requests to process native ESI to load into Relativity for review

Create New Project

Customer ID: TJX01 - The TJX Companies, Inc

Master Project ID: TJX00_VERSA - Versata Software Inc vs Callidus Software Inc

Client Matter ID: TBD001

Client Facing Volume: VERSA_REV004

Job Type Code: RV - Native Export

Create

Figure 3. Create new project.

9. Click Create

10. You will be directed to JIRA, and the following fields will be auto-populated from Dynamics:

- FQ Job Code
- Client Name
- Matter Name
- Matter Number
- Client Facing Volume(s)

Create Issue

Enter the details of the issue...

Project: Lighthouse Production

Issue Type: Native Production

Create Job ProdOps Fields Obsolete Fields

* FQ Job Code: TJX00_VERSA_RV004
Fully Qualified Job Code
(ClientCode_MatterCode.JobCode)

Client Name: The TJX Companies, Inc
Client Name From CentralData, Referenced by FQ Job Code.

Matter Name: Versata Software Inc vs Callidus Software Inc
Matter Name From CentralData, Referenced by FQ Job Code.

Matter Number: TBD001
Matter Number pulled from CentralData based on FQ Job Code

Client Facing Volume(s): VERSA_REV004
Client Facing Volume(s) From CentralData, Referenced by FQ Job Code.

Figure 4. Create issue.

11. Select the appropriate JIRA component(s)



Figure 5. Select component.

Table 3

JIRA Components and Description

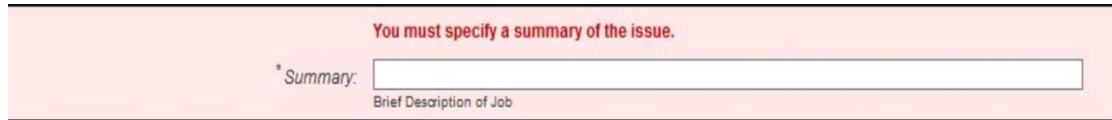
JIRA Component	Description
Accounting	Do not use
Coordination	(Rare)
Data Processing	All jobs related to Prod Ops (i.e., Nuix, eCapture) should include this component
Document Services	All jobs going through eCapture
Engineering	Do not use
Hosted Solutions	Jobs requiring the use of Relativity, Equivio, or Clearwell
Intake	Jobs requiring staging of data from a native source such as the ESI (new data)
Nuix	Jobs requiring processing in Nuix
Prod Ops Request	Do not use
Quality Assurance	All jobs involving QA should include this component (does not apply to HS tickets)
Rush Processing	Do not use
Sales	Do not use

12. Select the appropriate non-hourly billable components (NHBC)



Figure 6. Select NHBC.

13. Specify a summary or brief description of the job (this should match the Brief Summary section of the Work Order)



You must specify a summary of the issue.

* Summary:

Brief Description of Job

Figure 7. Specify job summary.

14. Deadline: If you have already huddled with a queue manager, enter the agreed upon deadline here; otherwise you can leave it blank for now
15. Ordered by: Input client/outside counsel contact that is associated with this request. This is helpful information for following up with the appropriate contact after the job has completed
16. Reporter: This should auto-populate with your username
17. Asset Number: If Prod Ops needs to stage data received from a client, input the assigned asset number here
18. Attachment: Attach all necessary reference documents (i.e., work order, track document, field list, cross-reference, etc.)
19. Security Level: Select Visible to Internal and External Users if the matter has been approved to be worked on by HCL; otherwise, select Visible to Internal Users
20. Type of Discussion:
- *New Matter, New Job*: This type of job will alert the queue manager to set up a huddle with a Prod Ops and HS resource
 - *Existing Matter, New Job (No Changes)*: This type of job does not

require a full huddle and can proceed with a short deadline huddle with the queue manager

- *Existing Matter, New Job (With Changes)*: This type of job will alert the queue manager to have a discussion with the TPM and determine whether a full huddle is required

21. Training Issue: This should default to No

22. Click Create

Deadline:

Overall Deadline When Ready For Final Check Must Be Reached

Ordered By:

Name of the Person at the Client's Office Who Ordered the Work

* Reporter:

Start typing to get a list of possible matches.
The PM or Sales Rep. Who Owns This Job (Set Other PMs on the Team As Watchers)

Asset Number:

Attachment:

The maximum file upload size is 10.00 MB. Please zip files larger than this.
Tech Spec, Billing Summary Sheet, Checklist, Etc.

Security Level:

Type of Discussion:

- None
- New Matter, New Job
- Existing Matter, New Job (No Changes)
- Existing Matter, New Job (With Changes)

Specify how huddles should be handled for this ticket.

Training Issue:

- None
- Yes
- No

Specifies whether or not this ticket is meant for training purposes.

Figure 8. Job creating page.

23. Add watchers to the job—this will ensure the pod (assignment group) and analyst receive JIRA emails when updates are made, and tickets show up in the appropriate filters:

- Click watchers



Figure 9. Watchers.

- Type in the last name or username of members of the pod and the analyst. As you type, suggestions should appear; select the appropriate usernames.

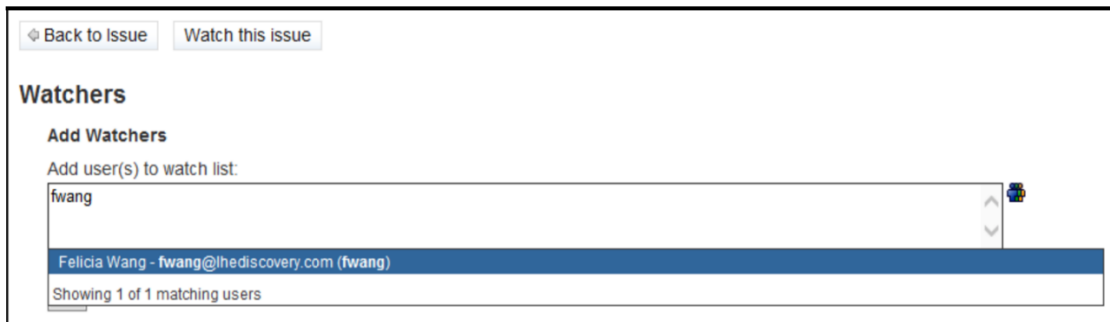


Figure 10. Adding watchers.

- Click Add
- Click Back to Issue to return to the JIRA ticket

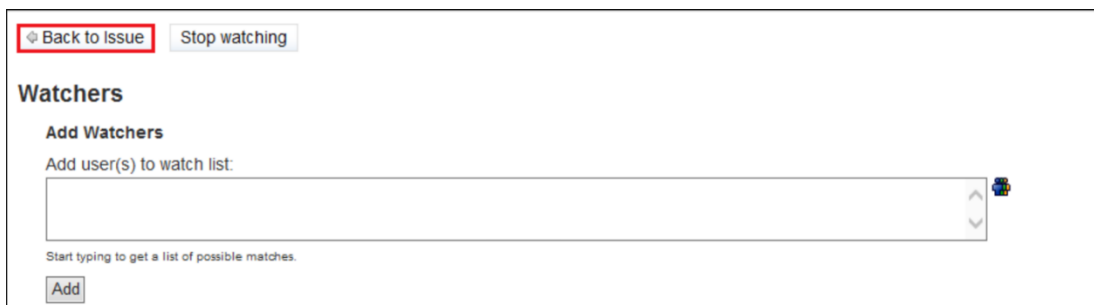


Figure 11. Back to issue.

24. Huddle with a queue manager and update the ticket with a deadline if one has not already been set.

Production WOR

General. The TPM huddles with the ProdOps supervisor on deadline.

JIRA.

- Ensure JIRA Ticket Summary matches WO Brief Summary and associated Work Order (WO) instructions:

SMART_PROD002 | Export Natives, Redacted Images, DAT | Image,

GENERAL INFORMATION

Client: GOOGLE LDT Key: LDT- 18201

Matter: SMARTFLASH New matter Ongoing matter

Volume: SMART_PROD002 PM: ALKI

FQ Job Code: GOO01_SMART_IR002 AM: JOSH STELLICK

Brief Summary: SMART_PROD002 | Export Natives, Redacted Images, DAT | Image, Endorse, Export out of eCapture | FTP | Overlay into Relativity

Figure 12. Work order (WO) general info.

- Ensure a deadline is entered

Dates

Created:	27/Jul/14 12:58 PM
Updated:	28/Jul/14 1:41 PM
Resolved:	27/Jul/14 8:50 PM
Deadline:	28/Jul/14 4:30 PM
Last Cmnt Tm:	2 weeks, 4 days ago

Figure 13. Specifying dates.

- Ensure the correct non-hourly billable components are selected:
 - PROD-IMAGE
 - PROD-BRAND

- SHIPPING (applicable only if burning and shipping media)

Main Data	ProdOps Fields	Billing Data	Accounting	Misc Status	Obsolete Fields
Non-Hourly Billable		PROD-IMAGE, PROD-BRAND, SHIPPING			
Components:					

Figure 14. Specifying NHBC.

- Watchers added (pod and analyst); click on "Watchers" to verify:

People

Assignee: Gurleen Sachar

Reporter: Erin Rubenstein

Watchers: 3 

Figure 15. Watchers added.

WO

General information.

- Ensure WO Brief Summary is updated to reflect current volume name as referenced in the appropriate tracking log and description includes top-level summary of steps to be performed
- Ensure LDT Key, Volume, and FQ Job Code have been updated as referenced in the appropriate tracking log
- All applicable files (i.e., field list, custodian cross-reference, etc.) are listed under Attachments and attached to ticket

Relativity export.

- Correct DB Name referenced
- Standard Production Relativity Export includes:
 - "Export DAT" (Preserve Unicode Characters): Include record count.

Note: If multi-page text files need to be exported for FPN records, please include that instruction here and reiterate that same instruction in the Relativity Export Notes (see example below)

(Hide) RELATIVITY EXPORT

 Change
 Section Notes/Strategy

DB Name: Lighthouse_Test_Workspace

Export CSV
 Export DAT
 Preserve Unicode Characters

Search Criteria/Saved Search Name:

<RELATIVITY SAVED SEARCH PATH> (517 records)
Please export MP text files for all records
 See **HS_Export_Field_List.txt** for fields to export

Export Redacted Images
 Export Other Images
 Export Native Files

Search Criteria/Saved Search Name:

<RELATIVITY SAVED SEARCH PATH> (517 records)

Relativity Export Notes:

Please export MP text files for all records.

Figure 16. Relativity export.

- "Export Native Files": Include record count. If applicable, this will include records where PROD_HOLD = <PRODVOL> AND Has Native = Yes

- "Export Redacted Images": Include record count and page count. If applicable, this will include records where PROD_HOLD = <PRODVOL> AND Markup Set - Primary = Has Redactions. Include both the document count and page count after the saved search path
 - "Export Other Images" (Hard Copy): Include record count and page count. If applicable, this will include records where PROD_HOLD = <PRODVOL> AND Has Native = No Include both the document count and page count after the saved search path. If the page count is greater than 1,000 pages/record OR the total page count is over 5,000, request a single-page TIF export with an OPT (multi-page TIF exports take longer). If the page count is reasonable, request a multi-page TIF export
- Field list with fields for Hosted Solutions (HS) to export is referenced within the Relativity Export Section Notes and includes our standard list of fields required to import into eCapture (see below) as well as any other fields needed for the production deliverable DAT:
- REVIEWID/BEGDOC
 - REVBEGATTACH/BEGATTACH
 - FILENAME
 - MD5 HASH
 - CUSTODIAN
 - FOLDER

- DOCEXT
- FILEDESC

Imaging and IQC.

- Imaging specifications match client expectations (i.e., production criteria)
- Any special Placeholder Options or Exception Handling instructions are specifically called out
- Redactions & Image Replacement Options
 - If SP Redacted TIFs were exported, check "Replace redacted images/text using Redaction Tool"
 - If MP Redacted TIFs or MP Hard Copy TIFs were exported, check "Dataset includes redacted TIFFs as natives."

Deliverables. A standard volume typically includes a Production build and a Relativity overlay build; for some clients, there is also an Adverse build.

- Ensure the production field list is referenced in the build and attached to the ticket
 - For the Relativity overlay build, our standard fields include: REVIEWID, PRODBEG, PRODEND, PRODBEGATT, PRODENDATT, PAGES, PRODVOL
 - If there are different sets of production fields in a single workspace (i.e., PRODBEG_ABC, PRODBEG_DEF), please ensure the Relativity overlay build fields are named as they are in Relativity.

- Correct Document File and Load File Formats selected
 - Production builds vary depending on what the client has agreed upon, but our standard Relativity build would include: Single-page TIFF/JPEG files, DAT (Preserve Unicode Characters), and an OPT
- If volume needs to be encrypted, a TC password is specified
- FTP or Deliverable Media instructions clearly outlined
 - FTP: Specify FTP URL, Username, and a path under "Additional FTP Instructions."
 - Copy to Media: Specify Quantity, Media Type, and Label Text

Literature Related to Problem

JIRA. JIRA is an issue and bug tracking project management tool developed by the Australian company, Atlassian (Atlassian documentation, n.d.). The base use of this tool is to track bugs in our software and mobile apps. But it is also used for project management.

JIRA issue. A JIRA issue would track a bug that underlies in the project. There are different types of issues (JIRA tutorial: A complete guide for beginners, n.d.).

Agile. We all know that the role of Project Manager seems almost impossible, with the ever-changing requirements (What is Agile project management?, n.d.). Hence we are moving away from the waterfall model to Agile methodology which allows the Project Managers to hit milestones and provide accurate project status reports. Agile management utilizes the method where small, usable segments of the

software product are specified, developed and tested in manageable, 2- to 4-week cycles (The Agile movement, 2008).

Agile with Scrum. This is where the responsibilities are shared among three roles (What Is Agile project management?, n.d.):

The Product Owner—handles goals, schedules, feature priorities

The Scrum Master—guides with task priorities

The Team Members—manages task assignment, quality control, and actual development

JIRA Software supports any agile project management methodology for software development, including Scrum.

Dynamics SL. Microsoft Dynamics SL is a business management solution that helps increase efficiency, the accuracy of billing and consumer satisfaction. Using Dynamics SL, we can manage our business with more flexibility. We can align our financials with our business processes.

Christine Weldon, Head of Finance, TWP Projects says, “*Everything we do for a customer is now stored in a single system. By having one repository for all this information in Microsoft Dynamics SL, we will be much more strategic in our approach and maximize the profitability of each project*” (TWP, 2009).

Literature Related to the Methodology

SERVICENOW. ServiceNow is a business to be studied. The company has grown exceptionally quickly. Since ServiceNow automates IT and other services with

cloud-based ITSM, we can access it anywhere from any system. Starting from Aspen, ServiceNow has now reached the Helsinki version, released this year.

Let's look at some concepts in ServiceNow. ServiceNow comprises of so much more than what's discussed here, but keeping the length of paper in consideration we'll stick to the basic OOB modules and the building blocks of customization. We'll study more about it during the implementation part of the paper.

IT service management. IT Service Management deals with having a single system for processes and infrastructure within the IT department and having a standardized solution all across IT.

Incident management. An incident is any disruption in service or a reduction in the quality of service, or the failure of a configuration item that has not yet affected a service, as defined by ITIL. The primary goal of Incident Management is to get the service back up running after an incident occurs. This ServiceNow module allows IT to capture these instances of service failure and to deal with them on the fly. Incident Management hugely helps the IT help desk to manage incidents more efficiently and promptly (ServiceNow | The Enterprise Cloud Company, n.d.).

Problem management. In the most basic terms, an incident that repeatedly occurs or to multiple users can be considered as a Problem. The primary goal of Problem Management is to investigate the cause of an incident through structured analysis, then document the solutions or workarounds, to prevent future service disruptions (ServiceNow | The Enterprise Cloud Company, n.d.).

Change management. Any Incident or Problem may lead to change. And any change may affect a multitude of configuration items and existing processes. Hence, every change needs to undergo impact analysis, risk analysis, approvals and result in the increased value to business. The concept of Change Management in ITIL also includes developing a business justification (ServiceNow | The Enterprise Cloud Company, n.d.).

Knowledge management. This module in ServiceNow captures all knowledge from across the organization making it available to use for all the employees. According to ServiceNow, *Knowledge Management is a key enabler in helping organizations solve problems faster and at a lower cost.* One aspect of Problem management is to document solutions and workarounds, and this is done in the knowledge base. So when a user calls the IT help desk regarding an issue, the person can use the Knowledge Base to see if this is a documented incident or a problem and provide a solution if one exists, and if not, keep the user informed of any development (ServiceNow | The Enterprise Cloud Company, n.d.).

Configuration and asset management. A business needs to know what assets are currently in their possession, to effectively manage and improve the systems. ServiceNow makes use of the CMDB, i.e., Configuration Management Database to keep track of all the configuration items within the organization. CMDB is highly useful for Change Management, so IT can know what configuration items will be affected by a change, or what configuration items have been affected by an

incident. The Asset Management deals with the financial aspect of the company's CIs (ServiceNow | The Enterprise Cloud Company, n.d.).

HR service management. Apart from IT, ServiceNow can be used for any department within the organization. Out of the box, ServiceNow provides functionality to support the HR applications, focus on and improve the business resources\.

(ServiceNow | The Enterprise Cloud Company, n.d.).

There are many other features in ServiceNow, but let's not go into too much detail.

ServiceNow platform. In ServiceNow, everything is managed as tables and forms (records). Let us take a look at some building blocks in ServiceNow.

Business rules. A business rule is a server side script that runs on a table when a record is inserted, updated, deleted, or queried, either before or after the fact, based on the condition set.

Client scripts. A client script is a client side (browser level) script that runs on the form/record, the 'when' is decided based on the conditions set.

UI/data policies. A UI policy also runs on the form level and is used to manage fields' visibility, write access and mandatory property. Data policy is a UI policy that runs when the record is imported from a third-party application.

Script includes. Script Include is also a server side script that runs only when called. They can be called multiple times from any table. It is a best practice to use Script Includes instead of Global Business rules.

ACLs. Access Control List is a list of access control rules on the tables that are evaluated at the server level and work both on forms and lists. These are the basic building blocks in ServiceNow; we will learn more about others as we go on.

Summary

In this chapter, we learned about the existing process as part of background related to the problem, then about the tools and techniques used in the existing process, and some part of the proposed system, its basic concepts, and building blocks. Going further, we'll learn how we use them to implement our customized solution. In the next chapter, we will learn about the methodology and what has been done so far.

Chapter III: Methodology

Introduction

In this chapter, we will learn about the tools and techniques used to implement the proposed solution and the hardware and software requirements.

Design of the Study

I would say that this study used a mix of both qualitative and quantitative approaches because the goal of this paper is to improve the performance of the existing system. And we can see the result of that in quantitative terms such as no. of staff hours reduced per job, the level of efficiency in maintaining the integrity and confidentiality of information, etc. At the same time, it is also said to be qualitative because the feedback is collected from the company once existing process is replaced by ServiceNow and those will be used for the results as well.

Data Collection

Data is collected from Product Documentation and ServiceNow Community posts and articles. Also, sample data sets are used from the personal developer instance provided to the user by ServiceNow.

Data Analysis

Tools and techniques. ServiceNow is the cloud-based ITSM tool used for this implementation. Apart from that, JIRA, Microsoft Dynamics SL, MS Excel, are used to study the existing process.

Hardware and software environment. Since ServiceNow is a cloud-based tool, let us see what browsers it supports:

Google Chrome, Mozilla Firefox, Internet Explorer, Microsoft Edge, Safari.

However, there are a few issues when using ServiceNow with IE.

MID Server is a Java application that runs as an intermediary between ServiceNow and other external applications. Since ServiceNow is a cloud application, for it to bypass any company's firewall and access information (e.g., for populating CMDB), a MID server needs to be installed at the company's location.

To install MID server, the system requirements are:

Minimum configurations:

- 4GB of available RAM per MID Server
- 2GHZ CPU, with a multicore CPU preferred
- 500MB of disk space per MID server

Supported systems:

The following systems support the MID Server.

- *Windows Server*

Service Mapping can discover Windows-based servers only using MID Servers installed on Windows Servers.

- Windows Server 2003
- All Windows Server 2008 and 2012 editions are supported.
- Virtual machines
- 64-bit systems

.NET Framework version 3.5 is required for Service Mapping support and Windows pattern-based discovery.

➤ Linux

- Virtual machines
- 64-bit systems

On 64-bit Linux systems, you must install the 32-bit GNU C Library (glibc). The installation command for CentOS is: `yum install glibc.i686` (MID server system requirements—docs.servicenow.com, n.d.).

Chapter IV: Data Presentation and Analysis

Introduction

In this chapter, we will explore the implemented design and customizations and the procedures used to develop the customizations.

Data Presentation

Let us first get an in-depth understand the concepts and building blocks in ServiceNow platform. We had a brief overview in the earlier chapters, but here, we will learn what comprises of each of those concepts which helps in the decision-making process for when and where to use them.

Server side Scripts:

Business rule. In a business rule, we need to specify what table it runs on, which application it is stored in, whether it is active, when it is supposed to run, under what conditions, etc. We also specify what is supposed to happen if the conditions apply.

If it is advanced, we use a script (JavaScript) to implement what needs to be. If it is not, we can set fields values or display a message on the specified table.

Let us see a few snapshots of a sample Business Rule and understand what each of the fields is meant for:

The screenshot shows the configuration page for a business rule named 'task events'. The interface includes the following elements:

- Name:** task events
- Table:** Task [task]
- Application:** Global
- Active:**
- Advanced:**
- When to run:**
 - When:** after
 - Order:** 100
 - Filter Conditions:** Add Filter Condition, Add "OR" Clause, -- choose field --, -- oper --, -- value --
 - Role conditions:** [edit icon]
- Actions:**
 - Insert:**
 - Update:**
 - Delete:**
 - Query:**
- Buttons:** Update, Delete
- Footer:** Versions, New, Go to, Created, Search, Update Versions, Created, Name, Source, State, Reverted from

Figure 17. Business rule—when to run.

Name: Name of the business rule.

Table: Table on which it runs. If global is selected, it runs on all tables.

Application: Application where the rule is stored. This is usually Global, unless we create a Scoped Application. This field is read-only and can only be managed via the application picker.

Active: If checked, the rule runs, otherwise stays dormant.

Advanced: Makes the script option available.

When to run: This section deals with the settings that decide when the rule should run.

When: There are four options here (advanced mode—relative to the database operation):

- a) before
- b) after

c) async

d) display

Insert: Executes the business rule when a record is inserted into the database.

Update: Executes the business rule when a record is updated in the database.

Delete: Executes the business rule when a record is deleted from the database.

Query: Executes the business rule when a table is queried.

Filter Conditions: Sets the conditions based on the field values in the table for the business rule to run.

Role conditions: This restricts the business rule to run for specific roles.

The screenshot displays the configuration interface for a business rule named "task events". The "Name" field is "task events" and the "Table" is "Task [task]". The "Application" is set to "Global". The "Active" and "Advanced" checkboxes are both checked. The "When to run" tab is selected, and the "Actions" sub-tab is active. In the "Actions" section, the "Set field values" field is set to "-- choose field --" and the "To" field is set to "-- value --". There are also "Add message" and "Abort action" checkboxes, both of which are unchecked. Below the "Actions" section, there are "Update" and "Delete" buttons. At the bottom of the interface, there is a "Versions" section with a "New" button and a "Go to" dropdown menu set to "Created". A search bar is also present. The bottom of the page shows a table with columns for "Created", "Name", "Source", "State", and "Reverted from", and a message "No records to display".

Figure 18. Business rule–actions.

The Actions section deals with the steps to happen if the conditions are satisfied.

Set field values: Sets the field values in the table. The first choicelist is the list of fields present in the table. The second one consists of operators (to, same as, to (dynamic)). The third one is the actual value to be set.

Add message: Apart from setting field values, we can also display info messages on the form if conditions apply. Choosing this will display another field called message, where we will specify the message to be displayed.

Abort action: We can use this checkbox to stop the current database action under specified conditions. This cannot be used in conjunction with setting the field values. Only displaying a message can be done along with this.

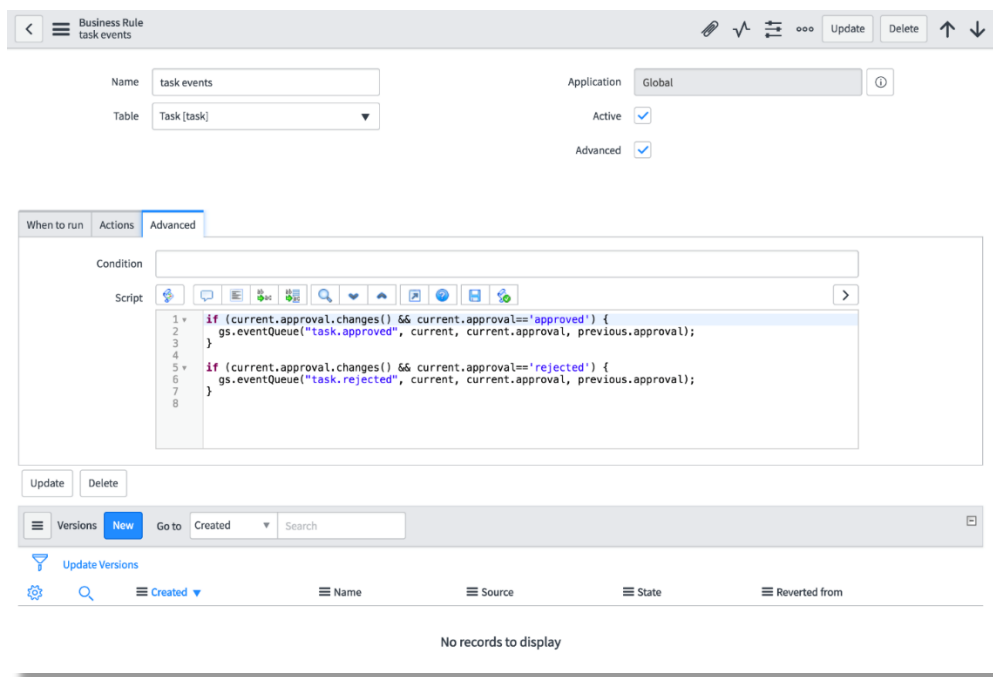


Figure 19. Business rule–advanced.

The advanced section only shows up when the advanced checkbox is ticked. Condition: Apart from using field values to set a filter condition, we can use this field to build a condition using script.

Script: The system auto populates this field with a function name that matches the When value.

onAfter for after, onAsync for async, onBefore for before, onDisplay for display. Note that the function name must match the when value.

Versions: This is a related list that's present in all configuration files for version management. Every time a record is saved, a version of it is created, so the older versions are retained.

Script include. This is another server side script, but unlike a business rule, it is not restricted to a specific table. It can be invoked from any table, works a little like Global Business rule. It is not recommended to use a Global Business Rule, because it drains performance. Instead of that, we could use a Script Include that only runs when invoked, not for every database action.

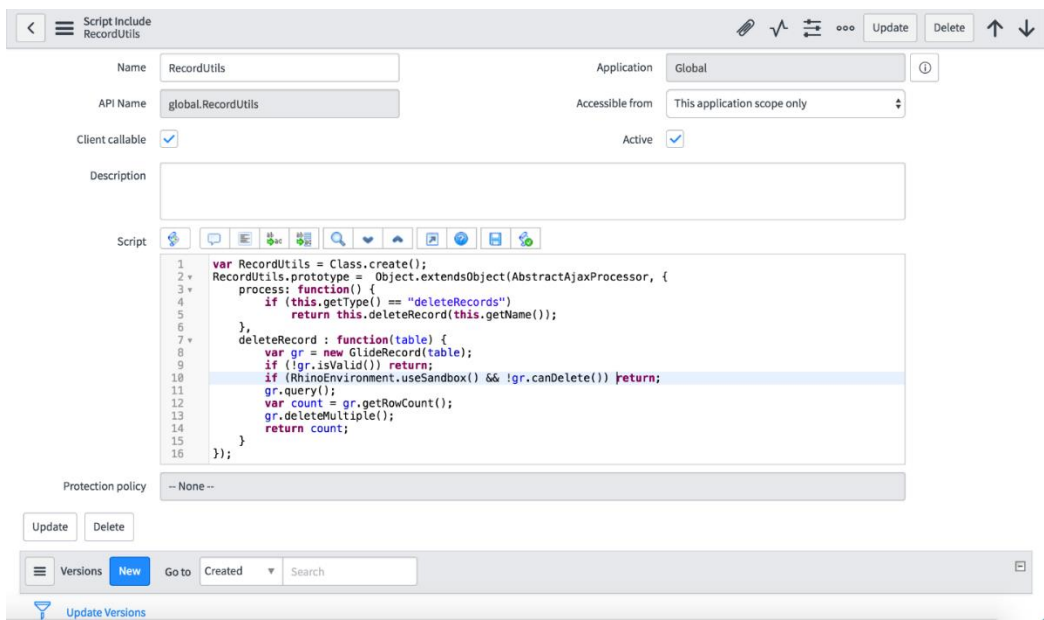


Figure 20. Script include.

Name: Name of the Script Include—we use this to invoke the script include. If we are defining a class, the name of the Script Include must match the name of the class, prototype, and type. If we are using a classless (on-demand) script include, the name must match the function name.

API Name: It is built by application scope.name of the script include.

Application: The application where this script include resides.

Accessible from: This decides which applications have access to this, whether it is only current application scope or all applications.

Client callable: If checked, it can be accessed from client side functions, like client scripts, list/report filters, reference qualifiers. If not, it is only accessible from server side scripts like business rules or other script includes.

Active: Decides whether the script is enabled or disabled.

Description: We can give a brief description of what the script include is meant for.

Script: Here we define the script to be run. The class name or the function name must match the name of the Script Include.

Protection Policy: This decides the level of protection for the script include. If none, it is open to everyone. Other two options are: Read-only and protected. Read-only prevents modifications to the script, while protected prevents users from even seeing the script, they can only use it.

Versions: Related list that stores the various versions of the Script Include.

How to use a Script Include. The format of a Script Include looks like this:


```

1  var NewScript = Class.create();
2  NewScript.prototype = {
3      initialize: function() {
4      },
5
6      myFunction: function(args){
7
8          //Put function code here
9
10     },
11     type: 'NewScript'
12 };

```

Figure 21. Script include–structure.

To invoke: (from a server script–invoking from client side will be explained later)

```

1  var func =new NewScript();
2  func.myFunction();|

```

Figure 22. Script include–call.

Script action. These are another type of server side scripts that are triggered by events.

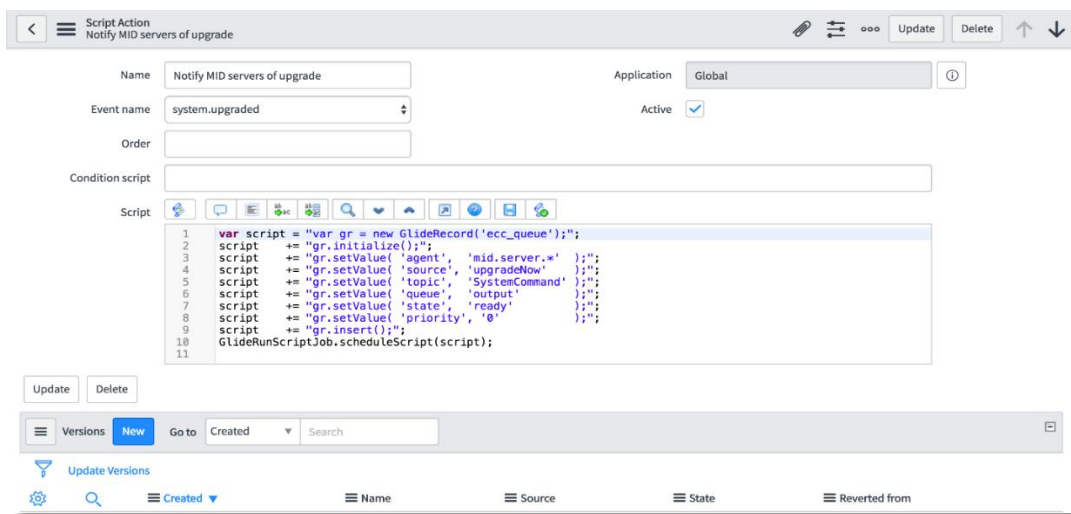


Figure 23. Script action.

Name: Name of the Script action.

Event name: Event that triggers the script action.

Order: Order in which the script will be executed. Helps if there are multiple script actions triggering with the same event.

Application: Application where which the Script action resides.

Active: Determines whether the script action is enabled or disabled.

Condition script: Can be used to build a scripted condition that should be evaluated to true for the script to be executed.

Script: Script to be executed. Two objects available to this type of script are event (that triggers the current script action) and current.

Client side scripts. Scripts that run on the client side (web browser level) are considered as client scripts. These are behaviors that occur during form events like page loading, form submission or field value change.

These scripts are basically transferred to the browser to run there instead of the server and that is why it is recommended to reduce the amount of client side programming whenever possible for performance reasons.

It must be noted that any manipulations done in client scripts apply only to forms. That means, list view of any table does not trigger the client scripts, so any manipulations done to fields in the client side must be in conjunction with similar manipulations to a list.

Client scripts. These are the client side scripts that run at the form level.

There are 4 types of client scripts:

- a) `onLoad`: This type of client script is executed when the form is loaded, before the control is given over to the user. It consists of a function `onLoad()`.
- b) `onSubmit`: This type of script runs when the form is submitted which can prevent submission of the function `onSubmit()` returns false.
- c) `onChange`: This type of script runs when a field value is changed. The deciding field should be specified. The `onChange()` function takes the following parameters:
 - i. `control`: the element which changed.
 - ii. `oldValue`: old value of the field before changing.
 - iii. `newValue`: value of the field after changing
 - iv. `isLoading`: specifies whether the field change is part of form loading.
 - v. `isTemplate`: specified whether the field change is part of template application.
- d) `onCellEdit`: This type of script runs when a cell is edited from the list editor. Its function `onCellEdit()` takes the following parameters:
 - i. `sysIDs`: array of sys id's of the fields being edited. Sys id is the unique identifier for a record in ServiceNow.
 - ii. `table`: table of the items being updated.
 - iii. `oldValues`: values of the cells being updated, before the change
 - iv. `newValue`: values of the cells being updated, after the change

- v. callback: a callback that continues the execution of other cell-related scripts.

The screenshot shows a web-based configuration interface for a client script. The title is 'Client Script' and the subtitle is 'Make Mandatory'. The interface includes several form fields and checkboxes:

- Name:** Make Mandatory
- Application:** Global
- Table:** Incident [Incident]
- UI Type:** Desktop
- Type:** onChange
- Field name:** Priority
- Active:**
- Inherited:**
- Global:**

Below these fields are text areas for 'Description' and 'Messages'. A 'Script' section contains a code editor with the following JavaScript code:

```

1  function onChange(control, oldValue, newValue) {
2      if (oldValue == newValue)
3          return;
4
5      g_form.setMandatory('short_description', true);
6
7  }

```

At the bottom of the form are 'Update' and 'Delete' buttons. Below the form is a 'Versions' section with a 'New' button, a 'Go to' dropdown set to 'Created', and a search field. The bottom of the interface shows a table with columns for 'Created', 'Name', 'Source', 'State', and 'Reverted from'.

Figure 24. Client script.

Name: Name of the client script

Application: Application where the client script resides.

Table: Table on which the client script runs.

Active: Specifies whether the script is enabled or disabled.

UI type: Specified which UI the script should run on—desktop/mobile/both.

Inherited: When checked, the client script also applies to any table that extends the table this originally runs on.

Type: Type of the client script—onLoad/onChange/onSubmit/onCellEdit

Global: If checked, the script runs on all views of the table, else, only runs on specific view, decided by the field 'View' which only shows up when 'Global' is unchecked.

Field name: If the type of client script is onChange, then this field specified what field on the table the script should be dependent on.

Description: Brief description of what the script is meant for can be given here

Script: Script to be executed.

Versions: Related list that stores the versions of the current client script.

UI scripts. UI scripts are similar to Script Includes. They provide reusable scripts to client side scripts, just like how Script Includes are reusable server scripts.

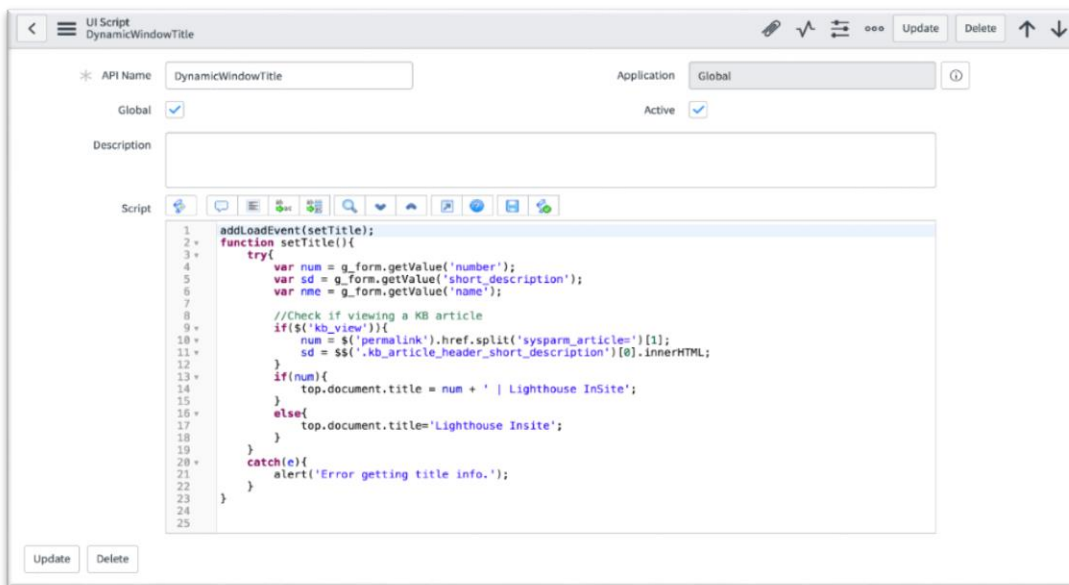


Figure 25. UI script.

API Name: Name of the UI script prefixed with the scope.

Application: Application where the UI script resides.

Global: When checked, the script loads on every page, else it should be invoked by a script.

Active: Determines whether the script is enabled or disabled.

Description: Brief description of what the script is supposed to do.

Script: Script to be executed.

Catalog client scripts. Client scripts work on regular forms. But for scripts to be run on Catalog items, this class of client scripts should be created.

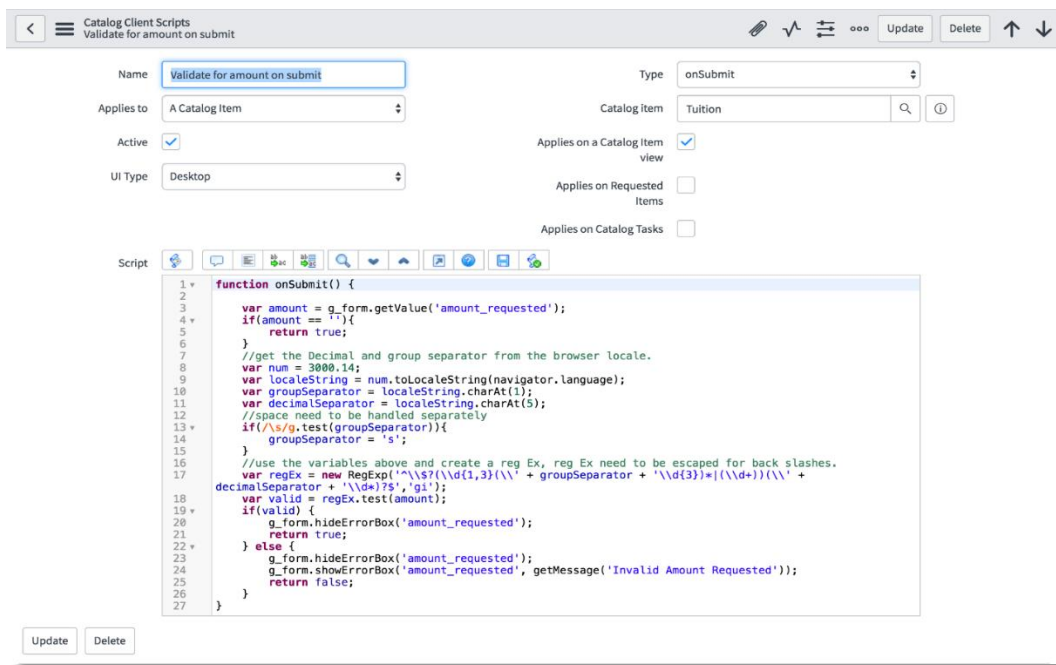


Figure 26. Catalog client script.

Catalog client scripts are very much similar normal client scripts, with a few field exceptions:

Applies to: Can be a catalog item or a variable set

Catalog item: Specifies the catalog item the scripts applies to.

Applies on a Catalog item view: Checking this applies the script to the catalog items present in the order screen of the Service Catalog.

Applies to Requested items: Checking this applies the script to the item form after it is ordered.

Applies on Catalog Tasks: Checking this applies the script when a Catalog Task form for the item requested is being shown.

Form administration. Apart from the form layout, annotations, form sections and templates among other things, we have UI actions and UI policies that play a major role in customization.

UI actions. The buttons, links, and context menu items that we see on forms and lists are defined by UI actions. We can create UI actions and use their scripts to define custom functionality.

Using UI actions, we can provide various controls: form button, list or form context menu item, list or form related link, list banner button, action choice list item.

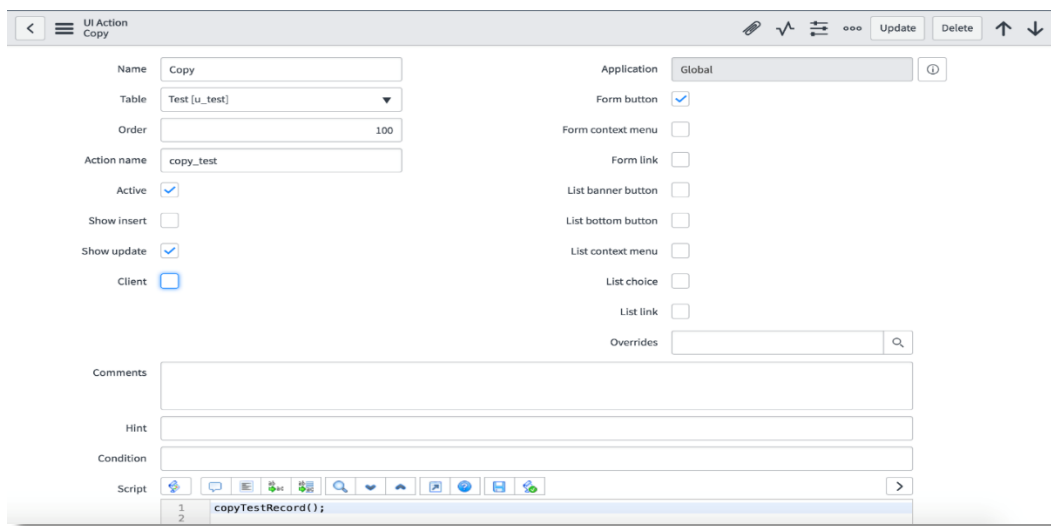


Figure 27. UI action.

Name: Name of the UI action (which is displayed on the button or as the text for a link)

Table: Table which the UI action appears in/applies to.

Order: Order of appearance of the UI action, relative to other UI actions on the same table.

Action name: Name of the UI action when referenced in other scripts, which must be unique.

Active: Enables or disables it.

Show insert: When checked, displays the UI action on new records that have not been inserted into the database.

Show update: When checked, displays the UI action on existing records.

Client: Script in the UI action, by default, is a server side script, unless this checkbox is checked, then it runs the script on the browser level. Checking this shows another field 'onClick' where we should specify the name of the function in the client script.

The checkboxes on the right column each specify how the UI action is rendered on the form or the list.

Overrides: Specifies the UI action that the current one overrides.

Comments: Brief description of the current UI action.

Hint: Specifies the text that appears when the user hovers the mouse pointer over the UI action.

Condition: Scripted condition that determines the availability of the UI action.

Script: Script to be executed when the UI action is clicked.

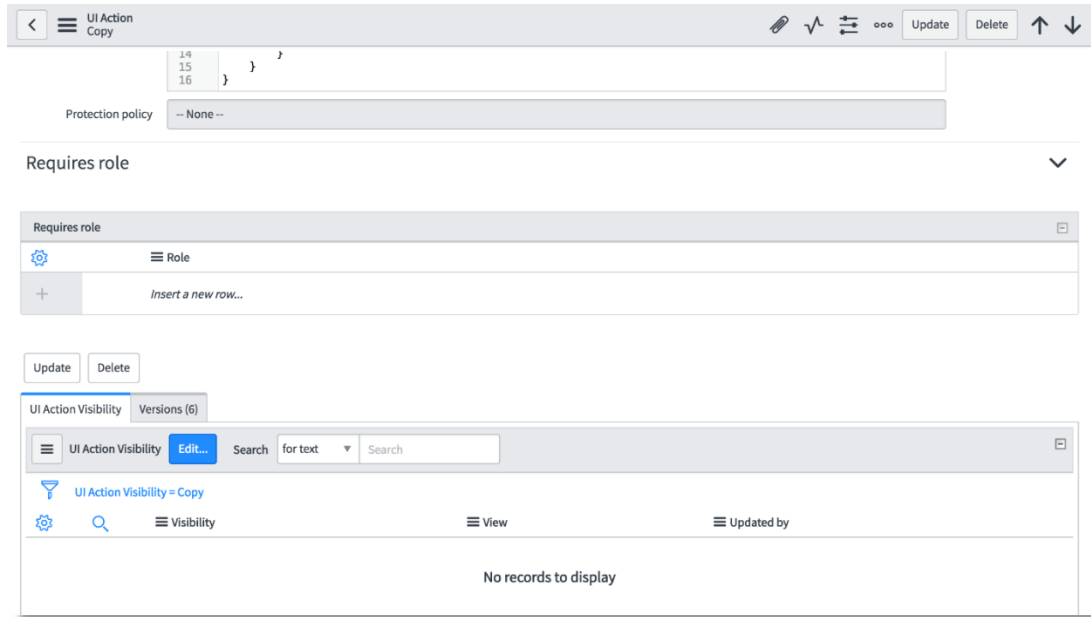


Figure 28. UI action.

Requires role: An embedded list where, the roles that this UI action should be restricted to can be specified.

UI action visibility: A related list that specifies what views the UI action appears on.

UI policies. These are an alternative to client scripts for changing the field values on a form dynamically.

UI policies evaluate all the fields on the table whether they are visible on the form or not.

UI Policy - Hide parent when is empty [Advanced view*]

Table: Application:

Active:

* Short description:

Order:

When to Apply | Script

Conditions:

Global: On load:

Reverse if false: Inherit:

Related Links
Default view

UI Policy Actions Search:

1 to 5 of 5

UI policy = Hide parent when is empty

Figure 29. UI policy—when to apply.

Table: Table which the UI policy runs on.

Application: Application where the UI policy resides.

Active: Enables or disables the UI policy.

Short Description: Brief description of what the UI policy does.

Order: Specifies the order in which the UI policies run, if there are conflicting ones.

When to Apply section deals with the conditions under which the UI policy is evaluated.

Conditions: Conditions that should be evaluated to true for the UI policy to be applied. They are evaluated when a field value is manually changed on the client side. Server side manipulations do not trigger the UI policies.

Global: Applies the policy to all views of the table.

Reverse if false: If the conditions satisfy, the UI policy's actions determine what should be done. If this is checked, then they are reversed.

On load: Usually UI policies work like onChange client scripts. If this is checked, then this policy is evaluated during the form load as well.

Inherit: When checked, the UI policy applies to the tables extended from the table this originally is applied to.

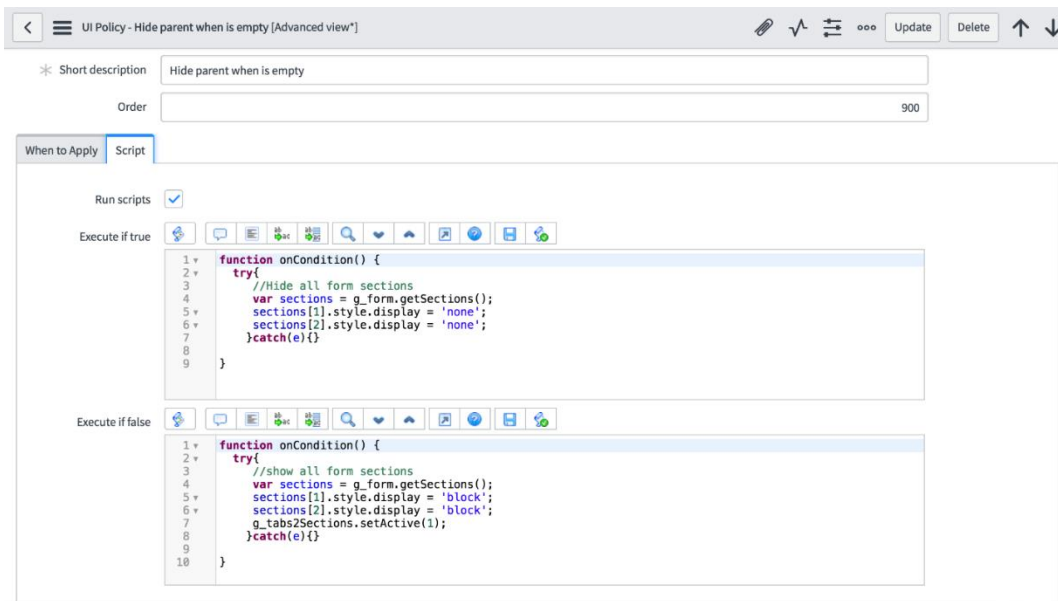


Figure 30. UI policy—script.

Script: This section is displayed in the Advanced view of the UI policy form.

Run Scripts: When checked, enables advanced scripted behavior for the UI policy.

Execute if true: Script that runs when the UI policy conditions evaluate to true.

Execute if false: Script that runs when the UI policy conditions evaluate to false.

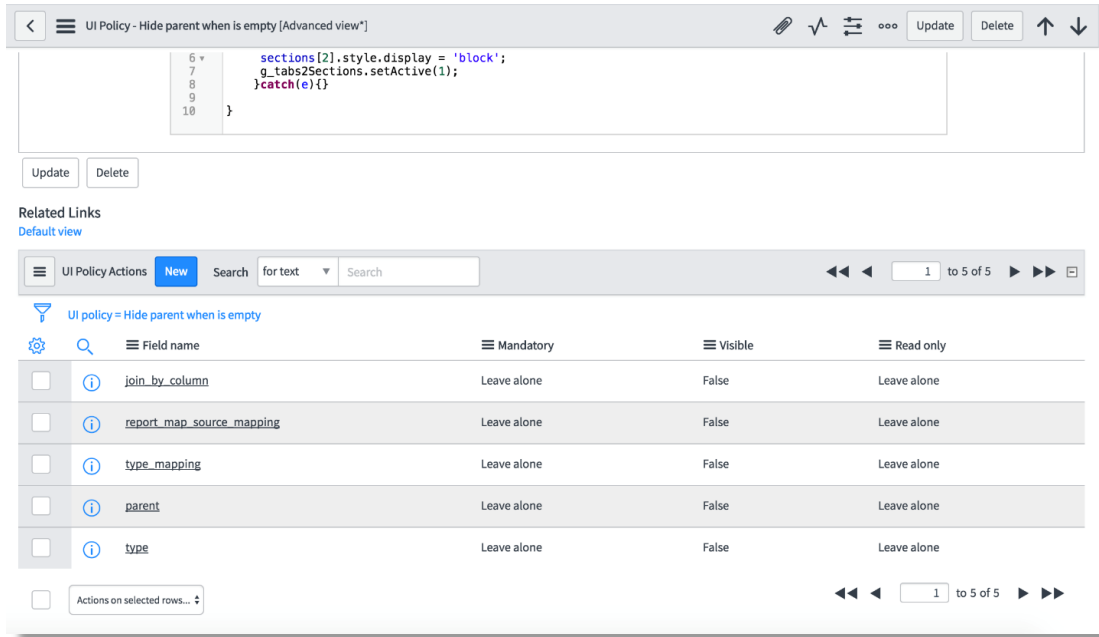


Figure 31. UI policy–UI policy actions.

UI Policy Actions: A related list that specifies what actions to take place when the UI policy conditions are satisfied (and reversed if not)

A UI policy action record looks like this:

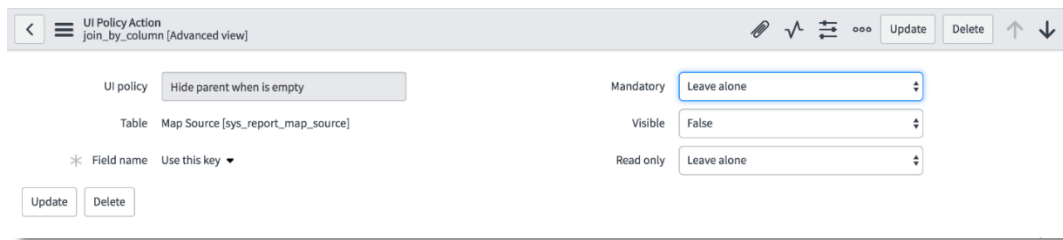


Figure 32. UI policy action.

UI policy: The UI policy that this action is related to.

Table: Table the action applies to.

Field name: Field on the table, that the UI policy action manipulates based on the UI policy conditions.

Mandatory: If true, sets the field as required.

Visible: If true, makes the field visible on the form.

Read only: If true, makes the field un-editable.

UI policies usually only apply on the form. But if we want the same policy to be applied when data is imported from external application, a Data policy must be created. An existing UI policy can be converted to a Data policy using the related link 'Convert this to Data Policy'.

Embedded lists vs related list. Earlier in this paper, I've referred to some lists as embedded lists and some as related lists. Let's understand the difference between them.

Both are basically another table's list view from inside a different table. Only when there is a relation between the two tables via a reference field, can they be displayed.

Embedded lists are within the current form and any changes made to the lists are only saved when the current form is saved. Related lists on the other hand, are not on the form, rather, they are attached to the current form at the end.

Changes made to these lists are automatically saved even if the current form is not submitted.

Embedded lists can be added to the form by 'Configuring the form layout'.

Related lists are added to the form by 'Configuring Related Lists'.

Platform administration:

Workflow. There are multiple items for platform administration, but workflows are the most important one to know about for this paper.

Workflows facilitates automating multi-step processes for the ServiceNow platform. Every workflow consists of a sequence of activities.

A graphical workflow editor is provided as part of the ServiceNow package, that represents the workflow, much like a regular flowchart.

A workflow is started when a triggering event occurs, like a new record being inserted, on when a record is updated with a certain value for a certain field.

It can even be triggered using a server side script.

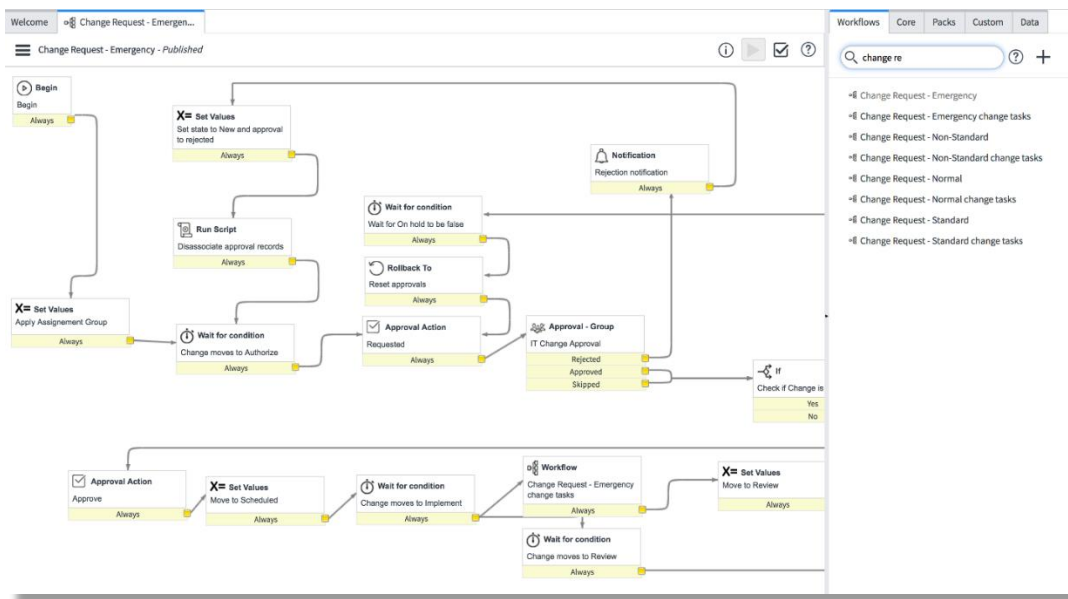


Figure 33. Sample graphical workflow editor.

The workflows column on the right menu, lists all the workflows in a ServiceNow instance.

The Core section lists all the available activities for designing a workflow.

The Custom section lists all the custom activities.

The Data section lists all the data elements.

Data Analysis

In ServiceNow, everything is made of tables. Even the business rules, client scripts etc., are all just tables, internally.

Since ServiceNow is a cloud based technology, the database resides on the ServiceNow server. Hence, we do not need to deal with maintaining the database, which will be done for us.

We have established that in this project, we deal with Clients, Matters and Jobs.

Let us look at the design we came up with for implementing the various Jobs and various Clients and Matters.

For Clients, we used an Out-Of-Box table, 'Company', which consists of all the fields corresponding to all the attributes that apply to a Company. Since, a client is a company in this project, we used the same.

For Matter, we created a custom table, extending from Task, an Out-Of-Box table that comes with the ServiceNow package.

Every Task record has a reference to the Company table which perfectly fits here, since every Matter is attributed to a Client.

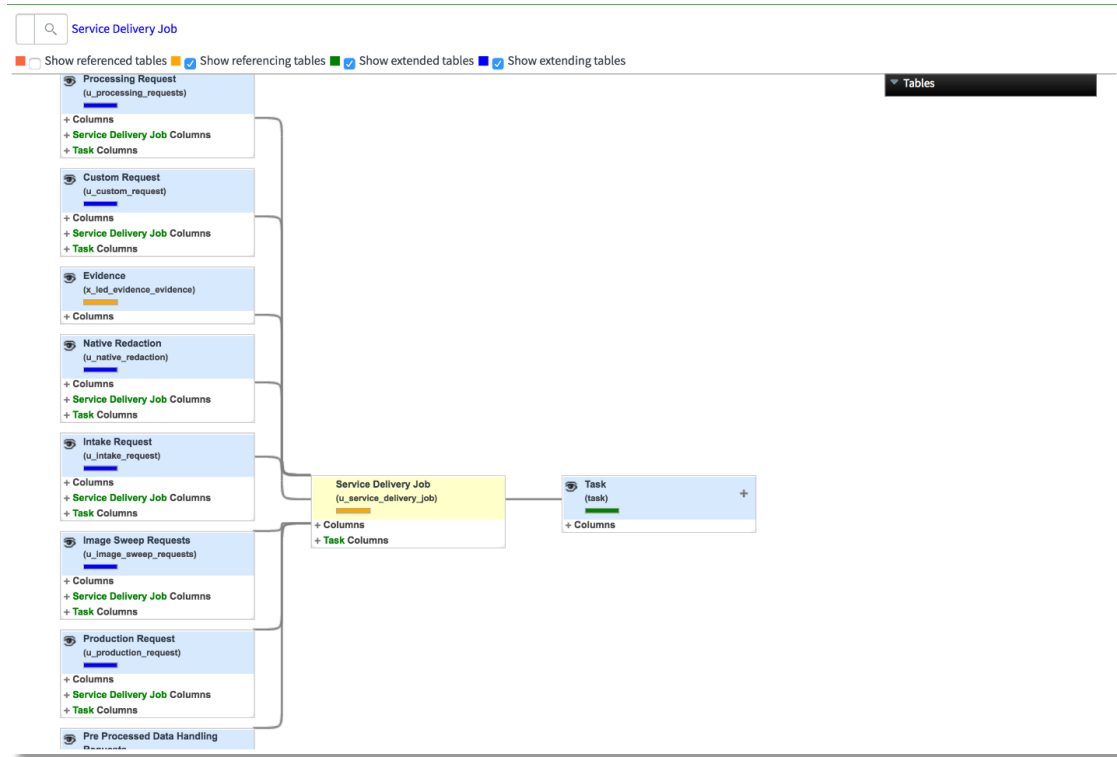


Figure 34. Database design for SD job.

Task is the closest resembling table to our requirements for a Job, hence that is the base table that the Service Delivery Job is based off on. Extending a table, also inherits all its configurations items like Business Rules, UI actions, Workflows etc., which reduces the amount of customization to a big extent.

Even in the different types of Jobs, we had some common fields and configurations to be built, so instead of redoing all of them for every table, we created a common table called 'Service Delivery Job' and placed all the common fields and configurations on this table.

Then we created all the other tables each corresponding to a job type. So, each of the job type tables consists of the fields from Task, as well as Service Delivery Job.

Similar to how Matters have reference to Clients, each of the Jobs has a reference to Matter and Client. The Client is restricted to what it is on the referenced Matter.

Now, we will move on to the workflows we designed for each of the Jobs and for Matters.

Obviously, for the data (existing records) imported from JIRA, no workflows will be applied since the process has already been started for them and it might be at any point within the process. So, the workflows will only be applied to newly created jobs or matters in ServiceNow.

New matter workflow. There is a process for taking on any new Matter.

Considering the existing process, we designed the workflow accordingly. We will look at the design and then look at the actual workflow implemented.

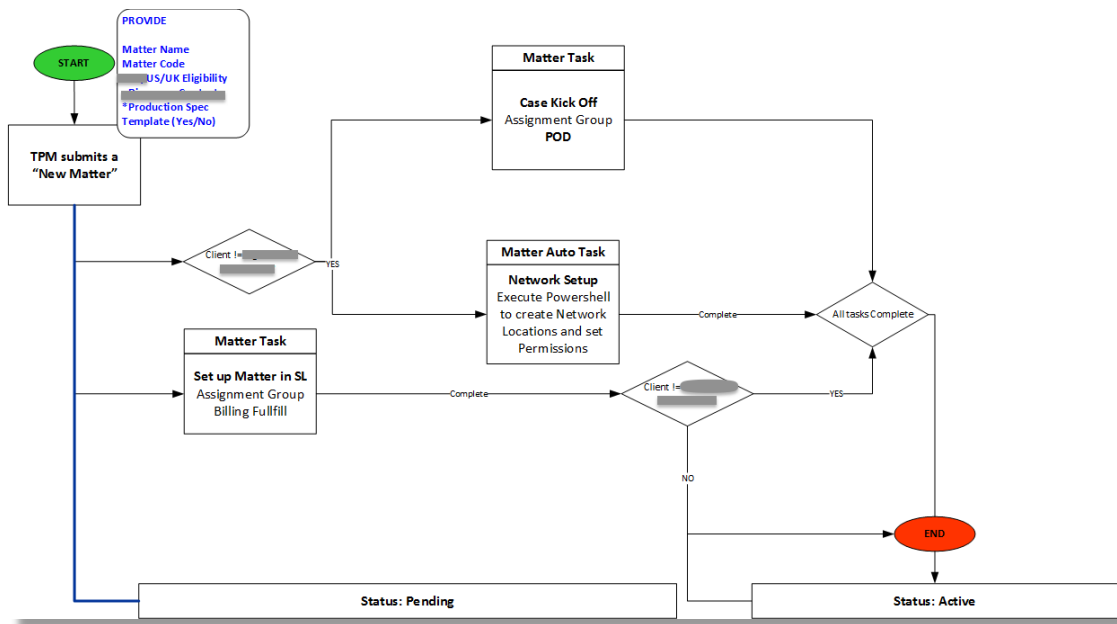


Figure 35. Workflow design for new matter.

The idea was to kick off the workflow and set the state of the Matter to Pending whenever a new Matter record is created in ServiceNow. The record is created by a Technical Project Manager (TPM) which denotes the first block.

The info in blue is only for reference purposes (not to be implemented). Then a task has to be created for the group 'Billing Fulfill' to set up the Matter in Dynamics SL. (We are moving away from SL to ServiceNow but until the project reaches to a relatively complete stage, we cannot sever ties with the other tools. Hence this task.)

And if the Client is an external client, then spawn two other tasks and once all three are complete (if all three exist) then set the state of the Matter to Active and finish the workflow.

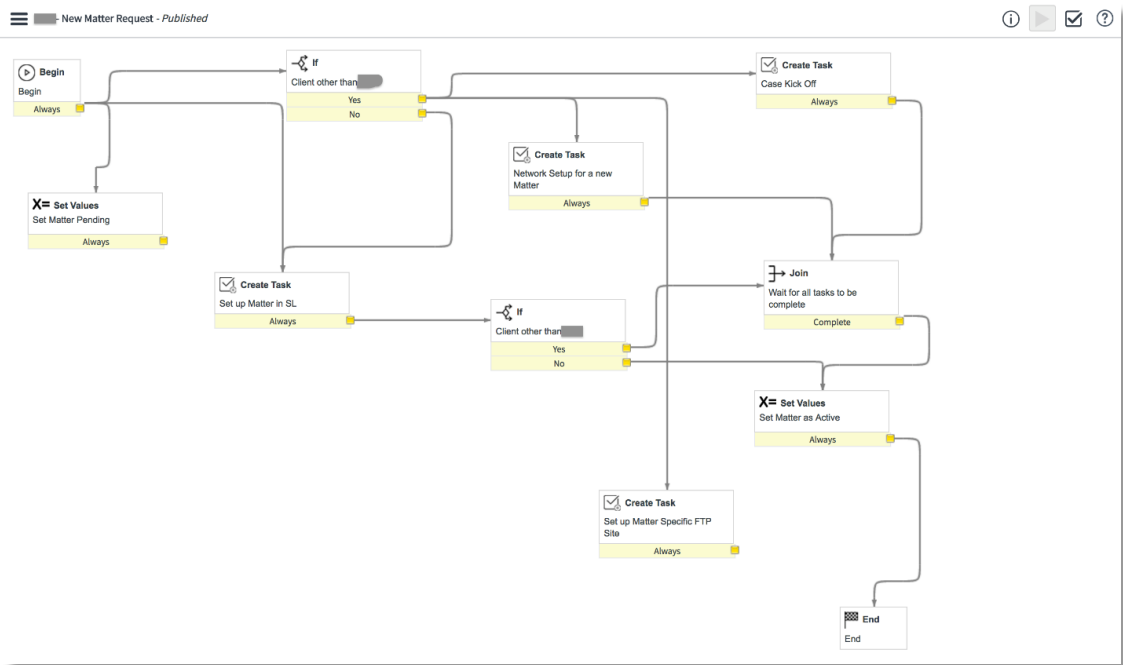


Figure 36. New matter request workflow.

Any new workflow created from a graphical editor, automatically adds the Begin and End blocks. The other activities between them should be created by us using the Core activities provided to us in the editor.

'Set Values' activity provides us the various fields from the table the workflow runs on, so we can set them to the values we like.

There's a 'Create Task' activity that creates records in the Task table or tables extending from Task. We use that to create a 'Set up Matter in SL' task, and based on the Client on the Matter, spawn two other tasks like designed in the workflow.

To keep tables and data contained, we created a table called 'Matter Task' extending from Task table that solely relates to the Matter table. All the tasks created in this workflow will be Matter tasks.

The 'Join' activity pauses the workflow until all the activities connected to it are completed.

Once all the tasks are completed, the Join activity is done and the workflow continues to set the Matter state to Active using 'Set Values' and then transitions to 'End' which finishes the workflow.

Let us see how each of the activities look like and what fields they consist of to help us understand the ServiceNow workflow better.

Activity Properties: Set Values

Workflow Activity
Set Matter Pending [Diagrammer view]

Name: Set Matter Pending

Stage: [Search]

Values

The Set Values activity sets the value of the fields specified below into the current record. [More Info](#)

Set these values	Pending
Status	Pending
-- choose field --	-- value --

Update

No templates are available [Create A New One?](#)

Figure 37. Workflow activity–set values.

Name: Name of the activity, a short description of what it does.

Stage: This field is for indicating the workflow progress. If an activity is active, then the stage set here will have the state as in progress. For the activities that are pending or complete, the stage shows the state accordingly.

Set these values: This assigns values to the fields in the table.

Activity Properties: If

Workflow Activity
Client other than [Diagrammer view]

Name: Client other than

Stage: [Search]

Conditions

The If activity checks a condition and/or script to determine which transition a workflow follows: 'yes' or 'no'. Use the 'Condition' builder to specify conditions that must be true for the activity to transition along the 'yes' path. [More Info](#)

Condition	is not
Client	[Redacted]

Check 'Advanced' to use a script for creating additional conditions. When you check 'Advanced', a text box appears where you can enter your script. In the script, add your code to determine when the script returns 'yes' or 'no'.

If the condition is false, or the script returns 'no', the activity transitions along the 'no' path.

Advanced

Figure 38. Workflow activity–if.

The 'If' activity provides conditional functionality based on the field values in the record. Alternatively, we can use the 'Advanced' field to create scripted conditions. When this checkbox is checked, a 'Script' field shows up which when evaluated to true, transitions the workflow through the 'Yes' path. If not, it takes the 'No' path.

The screenshot shows the 'Activity Properties: Create Task' dialog box. The title bar reads 'Activity Properties: Create Task' with a help icon. The main area is titled 'Workflow Activity' and 'Set up Matter in SL [Diagrammer view]'. It includes a 'Name' field with the value 'Set up Matter in SL', a 'Stage' field, and a 'Basics' section. The 'Basics' section contains a blue informational box stating: 'The Create Task activity creates a task for the current record. [More Info](#). The Priority will be set on the new task. Check Wait for completion if you want the workflow to pause until the task is complete. If you don't check Wait for completion, the task is created and the workflow proceeds.' Below this are three fields: 'Task type' (set to 'Matter Task [u_matter_task]'), 'Priority' (set to '-- None --'), and 'Wait for completion' (checked).

Figure 39. Workflow activity—create task—basics.

Name: Name of the Task activity—usually name of the task for easier understanding.

Task type: Type of the task to be created. Each of the table extended from the Task has a type.

Priority: Priority to be set on this task record.

Wait for completion: When checked, the workflow waits until the activity is completed, otherwise transitions forward.

Activity Properties: Create Task ?

Workflow Activity
Set up Matter in SL [Diagrammer view]

Populate task variables

In 'Task value from' specify the source of data for populating the fields in the new task. Select from Fields, Template, or Values. Once the source is selected, the appropriate fields will display.

Task values from: Fields

Fulfillment group: Finance Billing Fulfill

Assigned to:

* Short description: Set up the Matter in SL

Instructions:

Schedule

Figure 40. Workflow activity–create task–populate variables.

These fields are used to auto populate the fields on the created task.

Fulfillment group: Assignment group to be set.

Assigned to: User assigned to this task.

Short Description: Short description on the Task.

Instructions: Description on the Task.

Activity Properties: Create Task ?

Workflow Activity
Set up Matter in SL [Diagrammer view]

Schedule

Select how workflow determines the task's duration, due date, and schedule. In 'Time zone based on' specify how workflow should determine the timezone used to calculate task duration. Select 'no time zone' to use GMT. Once you've made a selection for 'Time zone based on', 'Due date based on', and 'Schedule based on', the appropriate fields will display.

Time zone based on: A time zone field

Schedule based on: (no schedule)

* Timezone field: Opened

Due date based on: A user specified duration

Duration: Days 0

Hours: 12 00 00

Script

Figure 41. Workflow activity–create task–schedule.

Time zone based on: Decides whether to use a field or static or no time zone at all.

Time zone field: Specifies which field's time zone is to be applied.

Due date based on: Specifies whether to use static duration set by us, or a field or a script.

Duration: If chosen to be set by us, we set the duration here.

Schedule based on: We can use schedule instead of continuous calculation of time.

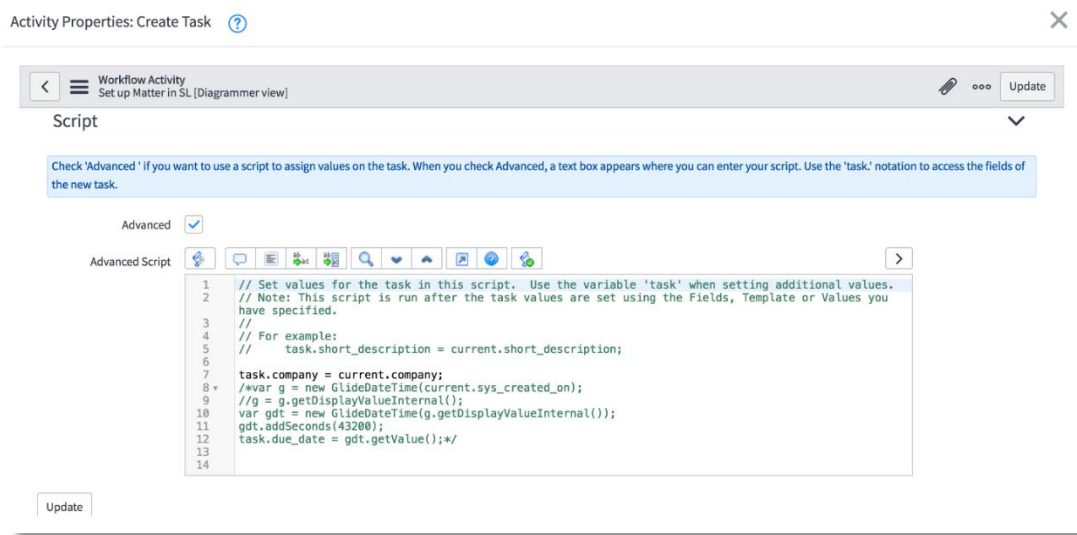


Figure 42. Workflow activity–create task–script.

Apart from setting the field values on the task using the fields shown above, we can also use the advanced script to assign values to the field.

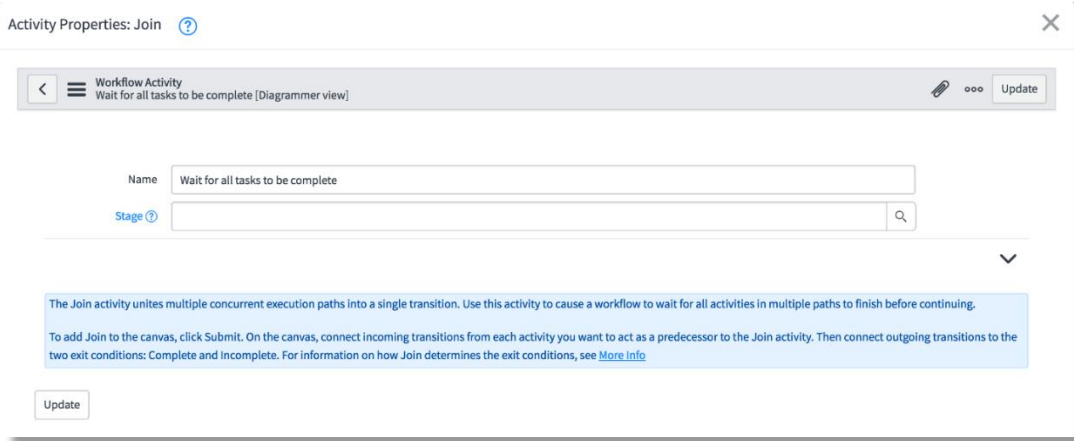


Figure 43. Workflow activity—join.

We name the Join activity the readability purpose but there's nothing to be filled in here. What's needed for this activity is the transition lines from the activities that need to be joined.

We have not yet designed or implemented any workflow for a new Client, but have for Jobs. Let us look at each of them.

Job initiation workflow. Like how all jobs have certain fields in common, there is some initial part of the process that is common for every job type. Since Service Delivery Job is a parent to all the job types, we created a common workflow for the job, then moved on to creating independent workflows for each job type. This 'Job Initiation' Workflow represents the initial process that is common to all jobs.

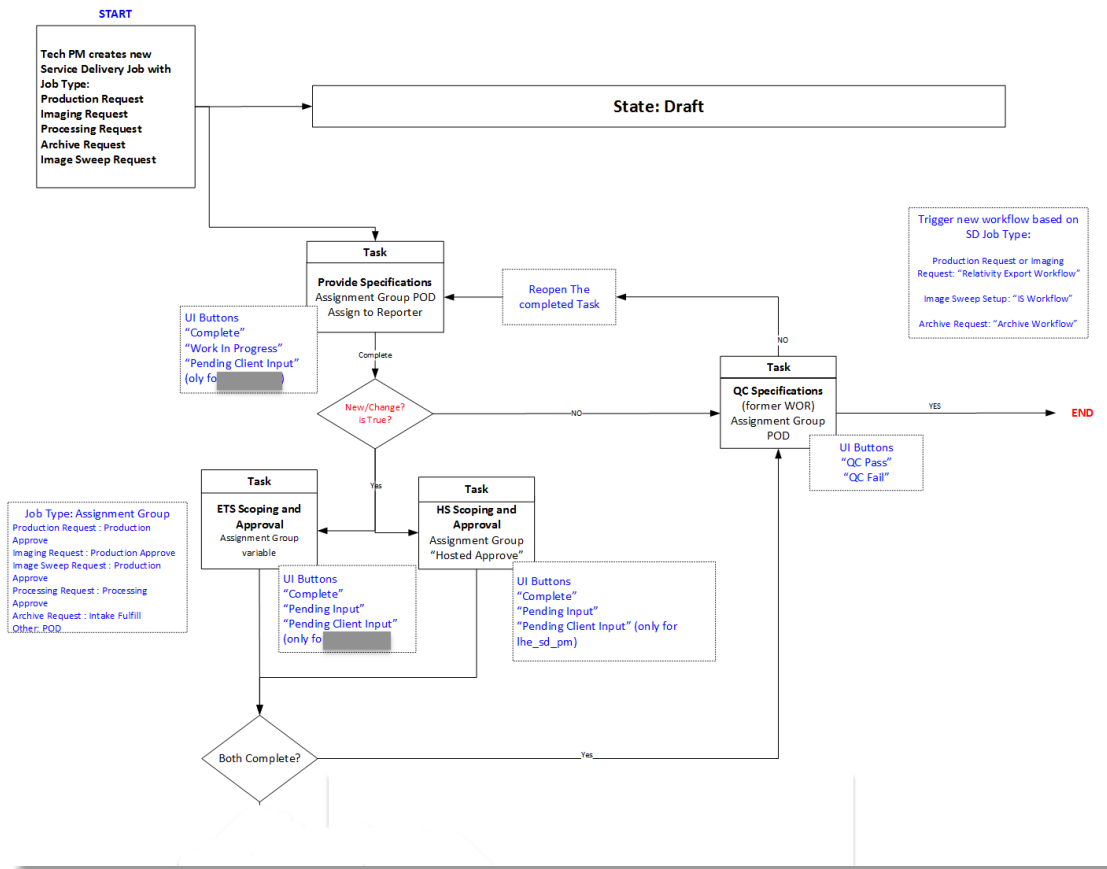


Figure 44. Workflow design for job initiation.

Idea was that TPM creates job which kicks off the Workflow. We'll see a detailed explanation for the workflow using the workflow implemented in ServiceNow.

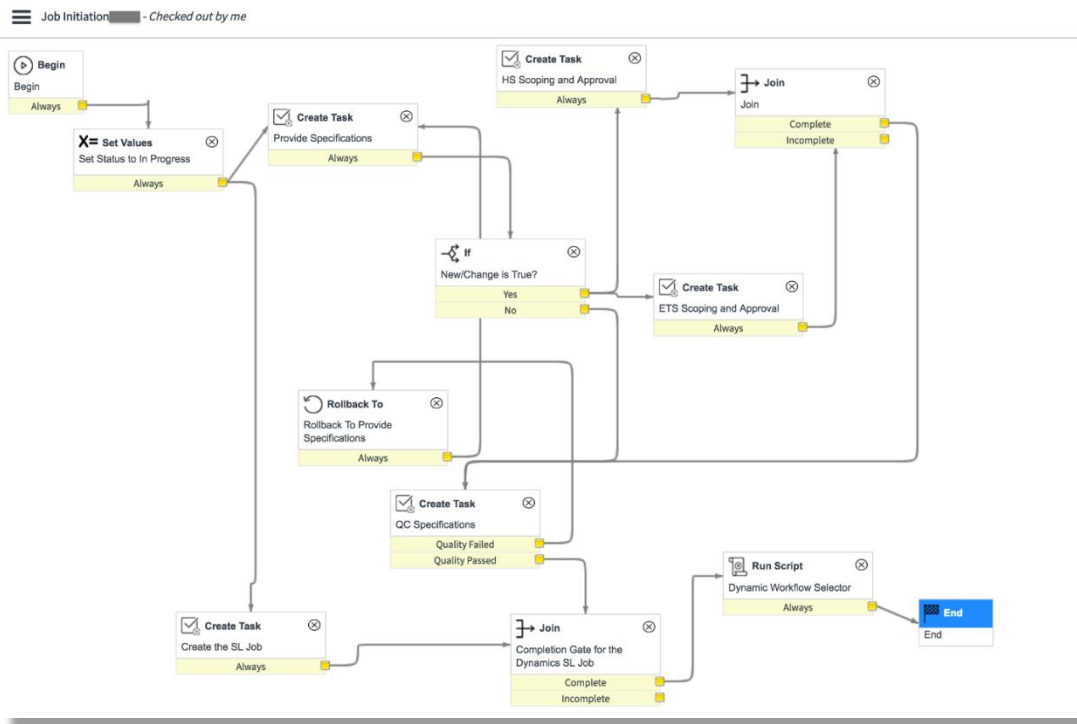


Figure 45. Job initiation workflow.

The 'Checked out by me' indicated that the workflow has been checked out. It means that all other users only have access to the workflow version that was saved before the checked-out time.

A workflow can only be edited when it's checked out. After the changes, we 'Publish' the workflow again to make it available to all users.

Once the workflow kicks off, the Job status is set to 'In Progress' and two tasks are created. Once for the TPM to provide Job specifications and another one that is automated which creates the job in Dynamics SL that corresponds to the ServiceNow job. We need this in place until the Dynamics SL is completely turned off and we have migrated away from it.

After that, based on the checkbox 'New/Change' the workflow takes different paths.

The 'Join' waits for both tasks connected to it to be completed before moving forward.

Once all the tasks are completed, the workflow waits for the Job creation in DL to be completed as well, before finishing the workflow.

The last activity 'Dynamic Workflow Selector' runs a script that triggers another workflow based on which job type it is.

Intake request workflow.

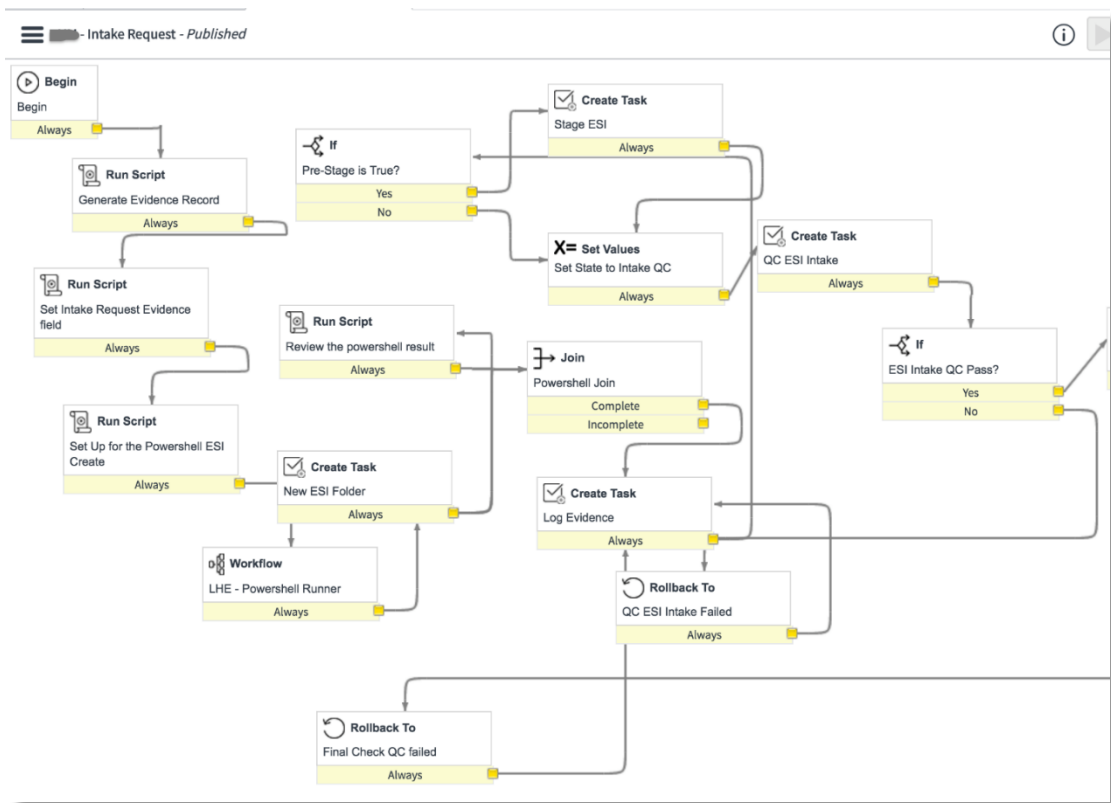


Figure 46. Intake request workflow.

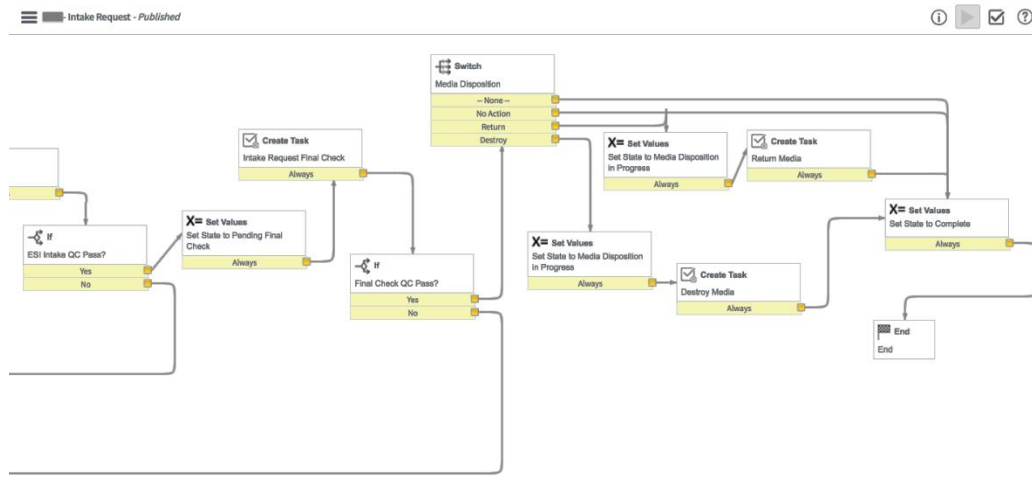


Figure 47. Intake request workflow.

This is the first workflow implemented for a job with job type = Intake Request. When the workflow kicks off, the run script activity creates a record in the ‘Evidence’ table and references that record in the current Intake Request. This is the script:

```

1  var gr = new GlideRecord('x_led_evidence_evidence');
2  gr.initialize();
3  gr.received_date = current.u_date_media_received;
4  gr.client = current.company;
5  gr.matter = current.parent;
6  gr.media_type = current.u_media_type;
7  //gr.u_description = current.short_description; //Edit: STRY0028974
8  gr.source = current.u_source;
9  gr.custodian = current.u_custodian;
10 gr.password = current.u_password;
11 gr.pm_log_notes = current.u_pm_log_notes;
12 gr.evidence_disposition = current.u_asset_disposition;
13 gr.jj_edr_number = current.u_jj_edr_medial_serial_number;
14 gr.linked_jobs = current.u_linked_jobs;
15 gr.received_from = current.u_received_from;
16 workflow.scratchpad.eid = gr.insert();
17
18 */var e = new GlideRecord('x_led_evidence_evidence');
19 e.addQuery('sys_id', eid);
20 e.query();
21 while(e.next()){
22     current.u_evidence = e.getDisplayValue();
23 }*/
24
  
```

Figure 48: Intake request workflow—generate evidence.

‘x_led_evidence_evidence’ is the name of the ‘Evidence’ table.

The ‘GlideRecord’ is a class that created an instance/record of the table passed as the parameter.

Initialize() function creates an empty record container with the fields present in the table.

Current is an object referring to the current Intake Request record.

Insert() function inserts the record created.

Workflow.scratchpad variable stores the value and is available all through the workflow.

The PowerShell script creates a local folder in the specified network location.

Then goes on to creating more tasks and setting values based on the fields in the Intake Request record. The 'Switch' functionality lets the workflow takes a different path based on the value of a choice field.

User can cancel an Intake Request which triggers another workflow called 'Intake Cancellation'. We implemented a feature to cancel any job type too but there's no workflow in place yet for them.

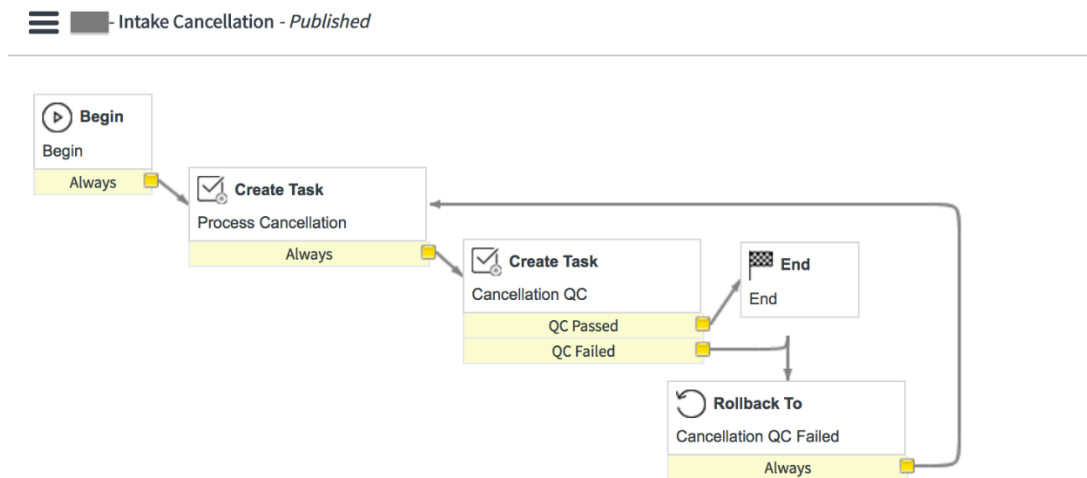


Figure 49. Intake request cancellation workflow.

First a 'Process Cancellation' task is created, upon its completion there is a Quality Check in place performed in the 'Cancellation QC' task. If the Quality check fails, the workflow rolls back to the first process cancellation task by reopening it. If the QC passes, the workflow ends, thereby successfully canceling the Intake Request.

Image sweep request workflow. This is the workflow implemented for the job type 'Image Sweep Request'.

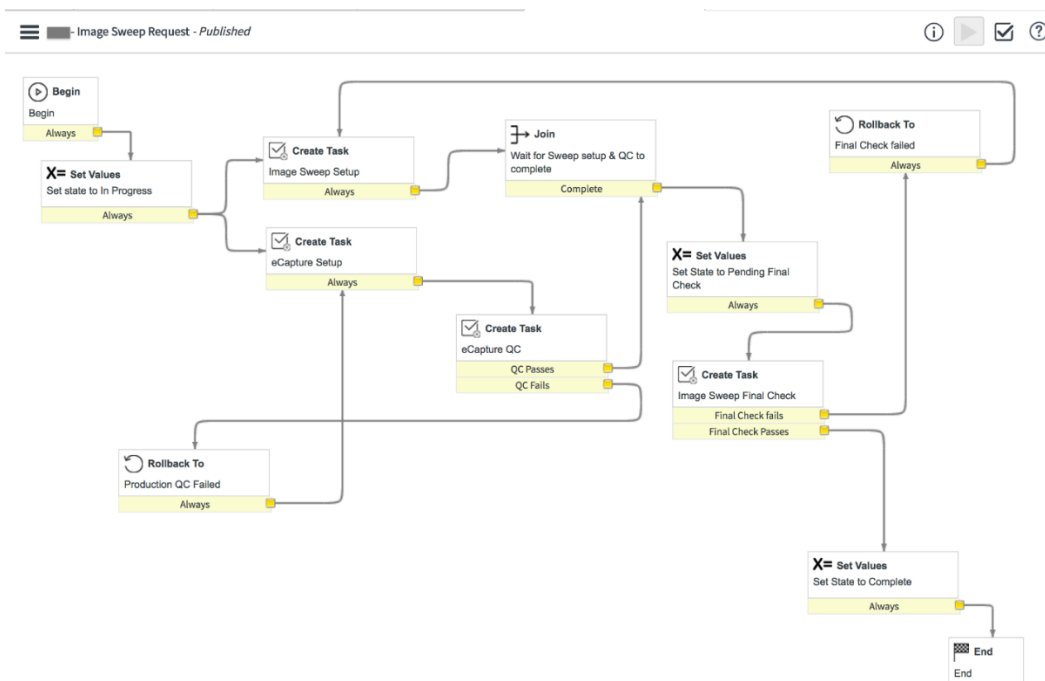


Figure 50. Image sweep request workflow.

This workflow is kicked off by the 'Dynamic Workflow Selector' run script activity in the Job Initiation workflow if the job type = Image Sweep Request.

The first activity is to set the state of the Request to 'In Progress', and creates two tasks 'Image Sweep Setup' and 'eCapture Setup' assigned to two different groups. 'eCapture Setup' has a Quality Check task associated to it.

Once the QC and other tasks are completed, the state of the Request is set to 'Pending Final Check' and created the task 'Image Sweep Final Check' for the PM team to give the final review of the job.

If the Final Check passed, the job is set to Complete and the workflow finishes.

Client archive request. This workflow is kicked off by the 'Dynamic Workflow Selector' run script activity in the Job Initiation workflow if the job type = Client Archive Request.

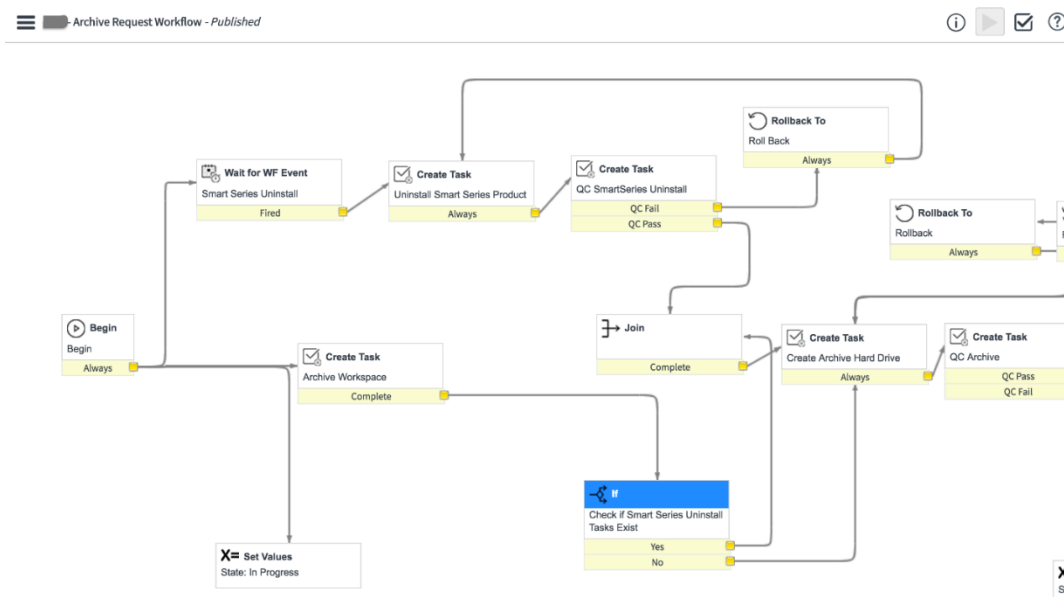


Figure 51. Client archive request workflow.

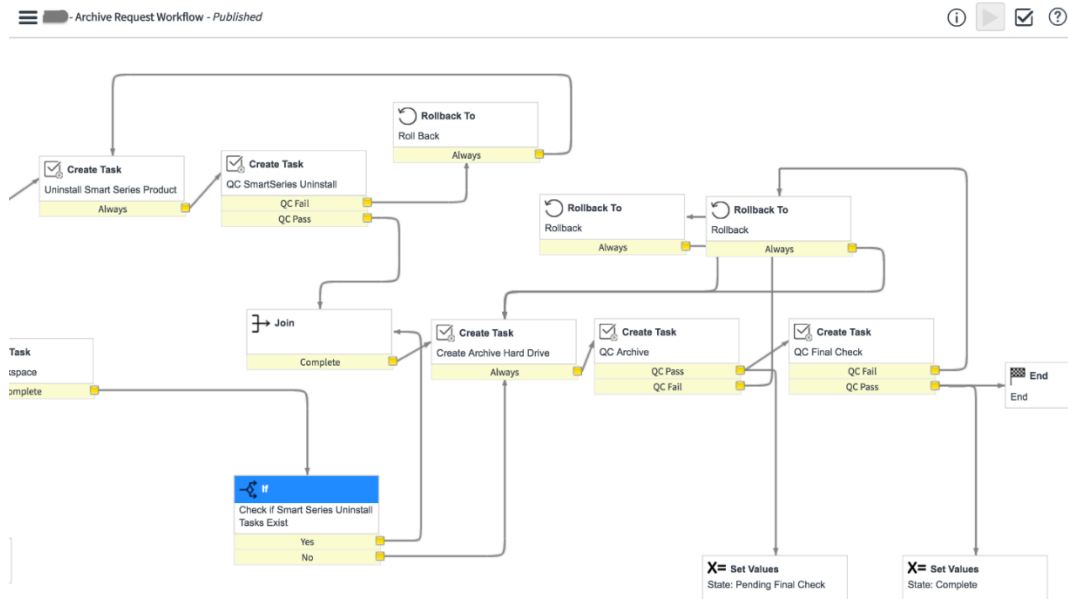


Figure 52. Client archive request workflow.

First step is to set the state of the Request to 'In Progress' and create a task 'Archive Workspace'.

The 'Wait for WF Event' waits until the event specified is fired from another activity in the Workflow.

Once fired, creates a task 'Uninstall SmartSeries Product' and performs a Quality Check.

The following tasks include 'Create Archive Hard Drive', 'QC Archive', and 'QC Final Check', while changing the state appropriately.

Business rules. Now, let's look at all the Business Rules we implemented at the Service Delivery Job level which runs on all the Job Types extend from it.

1) Set Status to Complete if Inactive

The screenshot shows the configuration for the business rule "Set Status to Complete if inactive". The rule is active and advanced. The "When to run" section is selected, showing the rule runs before insert or update operations. The filter conditions are: Active is false and Status is not Cancelled.

Business Rule: Set Status to Complete if inactive

Name: Set Status to Complete if inactive

Table: Service Delivery Job [u_service_delivery...]

Application: Global

Active:

Advanced:

When to run: before

Order: 850

Filter Conditions: Add Filter Condition Add "OR" Clause

All of these conditions must be met

Active is false

Status is not Cancelled

Role conditions:

Figure 53. Set status to complete if inactive—when to run.

2) Conditions specify that the rule is triggered before the insert or update of any job, if the active is set to false and status is not cancelled.

The screenshot shows the configuration for the business rule "Set Status to Complete if inactive". The rule is active and advanced. The "Actions" section is selected, showing the rule sets the status of the job to complete.

Business Rule: Set Status to Complete if inactive

Name: Set Status to Complete if inactive

Table: Service Delivery Job [u_service_delivery...]

Application: Global

Active:

Advanced:

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.

Set field values: Status To Complete

-- choose field -- To -- value --

Add message:

Abort action:

Figure 54. Set status to complete if inactive—actions.

If the conditions satisfy, the status of the Job is automatically set to 'Complete'.

3) Job Initiation Start

The screenshot shows the configuration for a Business Rule named 'Job Initiation Start'. The rule is set to be 'Global', 'Active', and 'Advanced'. The 'Table' is 'Service Delivery Job [u_service_delivery...'. The 'Advanced' tab is selected, showing a script that checks if the current table is in a list of tables defined in the 'lhe.job.include' property. If it is, it calls a workflow named 'Job Initiation'.

Business Rule Configuration:

- Name: Job Initiation Start
- Application: Global
- Table: Service Delivery Job [u_service_delivery...]
- Active:
- Advanced:

Script (Advanced Tab):

```

1 (function executeRule(current, previous /*null when async*/) {
2
3 // Add your code here
4 var tables, tableArray;
5 tableArray = new ArrayUtil();
6 tables = JSON.parse(gs.getProperty("lhe.job.include"));
7 tables = tableArray.ensureArray(tables);
8
9
10 if(parseInt(tableArray.indexOf(tables, current.sys_class_name.toString()) ) >= 0 ){
11     new lheWorkflowUtil("Job Initiation", true, {}, current);
12 }
13
14 })(current, previous);

```

Figure 55. Set status to complete if inactive–advanced.

This rule is triggered before any new Job Request is inserted.

It retrieves the values stored in the property 'lhe.job.include' which lists all the tables that the workflow should run on and loops through each of it using the array utility.

If the current table (retrieved by `current.sys_class_name.toString()`) is stored in the property, then it calls the Script Include which has the script that runs the workflow.

The parameters include, the name of the workflow to be called and the current record.

The screenshot shows a configuration window for a system property. The title bar reads 'System Property' and 'lhe.job.include'. On the right side of the title bar, there are icons for edit, refresh, and a menu, along with 'Update' and 'Delete' buttons. The main content area includes:

- Name:** A text input field containing 'lhe.job.include'.
- Application:** A dropdown menu set to 'Global'.
- Description:** A text area containing 'An array of tables to exclude, in job initiation.'
- Choices:** An empty text area.
- Type:** A dropdown menu set to 'string'.
- Value:** A text area containing '["u_image_sweep_request","u_production_request","u_client_archive_request"]'.
- Ignore cache:** An unchecked checkbox.
- Private:** An unchecked checkbox.
- Read roles:** A button with a pencil icon.
- Write roles:** A button with a pencil icon.

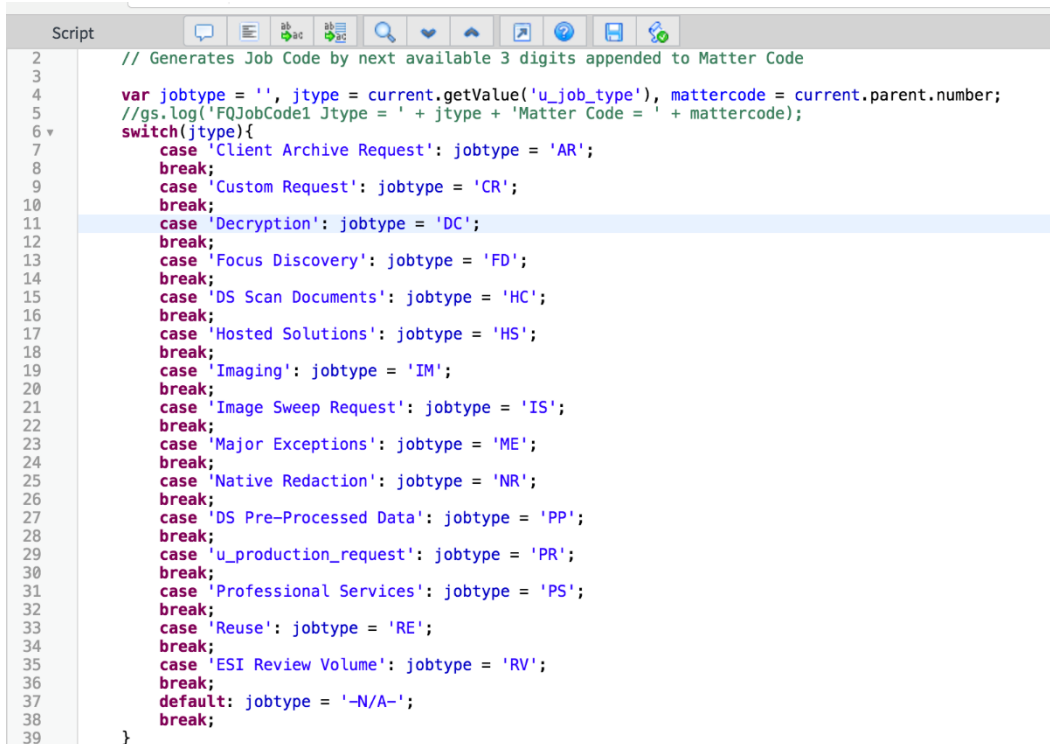
Figure 56. System property—lhe.job.include.

The names of the tables that the Job initiation should run on are stored in the property as shown above.

4) Generate FQ Job Code

This rule is implemented to create a unique code for each job that is computed by the Matter Code, followed by the Job type initial, followed by a unique number.

This is currently only implemented on Image Sweep Request but is created in such a way that it can be used for other jobs as well.



```

2 // Generates Job Code by next available 3 digits appended to Matter Code
3
4 var jobtype = '', jtype = current.getValue('u_job_type'), mattercode = current.parent.number;
5 //gs.log('FQJobCode1 Jtype = ' + jtype + 'Matter Code = ' + mattercode);
6 switch(jtype){
7   case 'Client Archive Request': jobtype = 'AR';
8     break;
9   case 'Custom Request': jobtype = 'CR';
10    break;
11   case 'Decryption': jobtype = 'DC';
12    break;
13   case 'Focus Discovery': jobtype = 'FD';
14    break;
15   case 'DS Scan Documents': jobtype = 'HC';
16    break;
17   case 'Hosted Solutions': jobtype = 'HS';
18    break;
19   case 'Imaging': jobtype = 'IM';
20    break;
21   case 'Image Sweep Request': jobtype = 'IS';
22    break;
23   case 'Major Exceptions': jobtype = 'ME';
24    break;
25   case 'Native Redaction': jobtype = 'NR';
26    break;
27   case 'DS Pre-Processed Data': jobtype = 'PP';
28    break;
29   case 'u_production_request': jobtype = 'PR';
30    break;
31   case 'Professional Services': jobtype = 'PS';
32    break;
33   case 'Reuse': jobtype = 'RE';
34    break;
35   case 'ESI Review Volume': jobtype = 'RV';
36    break;
37   default: jobtype = '-N/A-';
38   break;
39 }

```

Figure 57. Generate FQ job code.

For each job type, we have a unique initial defined using the Switch functionality.

Matter code is retrieved from the Matter reference field on the Request.

gs.log() generates a log record.

To generate the unique code, the process followed is:

- Query the job table for existing records with the same matter and same job type
- Then sort the records in descending order by Number.
- This will bring the highest numbered record at the top.
- Then we can slice the last 3 digits of the code and increment it by 1 to create the next available number and save it in the 'number' field.

```

39     }
40     //gs.log('FQJobCode2 Jobtype prefix = ' + jobtype);
41     var num, codestart = (mattercode+'_'+jobtype), gr = new GlideRecord('u_service_delivery_job');
42     //gs.log('FQJobCode3 codestart = ' + mattercode + '_' + jobtype);
43     //gr.addQuery('number', 'STARTSWITH', codestart);
44     gr.addQuery('parent', current.parent); //gets jobs with same matter
45     gr.addQuery('u_job_type', current.u_job_type); //gets job with same jobtype
46     gr.orderByDesc('number'); //sorts by desc order of number
47     gr.query();
48     if(gr.getRowCount() > 0){
49         gr.next(); //since order is desc, one next() will take us to the highest numbered job
50         num = parseInt(gr.number.slice(-3)) + 1;
51         //gs.log('FQJobCode4 last number = ' + gr.number.slice(-3));
52         //gs.log('FQJobCode5 next available number = ' + num);
53     }
54     else{
55         num = 1; //no records match, so start from 1.
56     }
57     var number = num + '';
58     while (number.length < 3)
59         number = '0' + number; // this is so the numbers with less than 3 digits have preceding 0s.
60     //gs.log('FQJobCode6 next available number after normalization = ' + number);
61
62     current.number = codestart+number;
63
64 })(current, previous);

```

Figure 58. Generate FQ job code.

5) Set Client

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. Use business rules to automatically change values in form fields when the specified conditions are met. [More Info](#)

Name	Set Client	Application	Global
Table	Service Delivery Job [u_service_delivery_j...]	Active	<input checked="" type="checkbox"/>
		Advanced	<input checked="" type="checkbox"/>

When to run: **Advanced**

Condition:

Script

```

1 * (function executeRule(current, previous /*null when async*/) {
2
3     //var client = new ServiceClientJob().getMatterClient(current.parent.toString());
4     //current.company = client;
5
6     current.company = current.parent.company.sys_id.toString();
7     current.assignment_group = current.parent.assignment_group.sys_id.toString();
8
9 })(current, previous);

```

Figure 59. Set client.

This rule is to ensure that all jobs consistently have the data for Client and assignment group. The client and assignment group in the current record is made consistent with that of in referenced Matter record.

6) Deactivate Metric

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. Use business rules to automatically change values in form fields when the specified conditions are met. [More info](#)

Name	<input type="text" value="Deactivate Metric"/>	Application	<input type="text" value="Metrics and Billing Management"/>
Table	Service Delivery Job [u_service_delivery_job]	Active	<input checked="" type="checkbox"/>
		Advanced	<input checked="" type="checkbox"/>

When to run: **Advanced**

Condition:

Script

```

1 (function executeRule(current, previous /*null when async*/) {
2
3   var gr = new GlideRecord('x_led_metrics_sd_metrics');
4   gr.addQuery('parent', current.sys_id);
5   gr.query();
6   while(gr.next()){
7     gr.active = false;
8     gr.update();
9   }
10
11 }(current, previous);

```

Figure 60. Deactivate metric.

This rule is triggered when a current job is deactivated.

It queries the metrics table to get all the metrics associated with the current request and set their active flag to false.

7) Calculate SD Metrics from job

Name	<input type="text" value="AIM - Calculate SD Metrics from Job"/>	Application	<input type="text" value="Global"/>
Table	Service Delivery Job [u_service_delivery_j...]	Active	<input checked="" type="checkbox"/>
		Advanced	<input checked="" type="checkbox"/>

Advanced

Condition:

Script

```

1 (function executeRule(current, previous /*null when async*/) {
2
3   var updateRate = new AIM_getMetricBilling();
4   updateRate.getSDBillingMetrics(current.sys_id, current.u_billable_component, current.correlation_display.toString(),
5   current.parent, current.u_job_type);
6 }
}(current, previous);

```

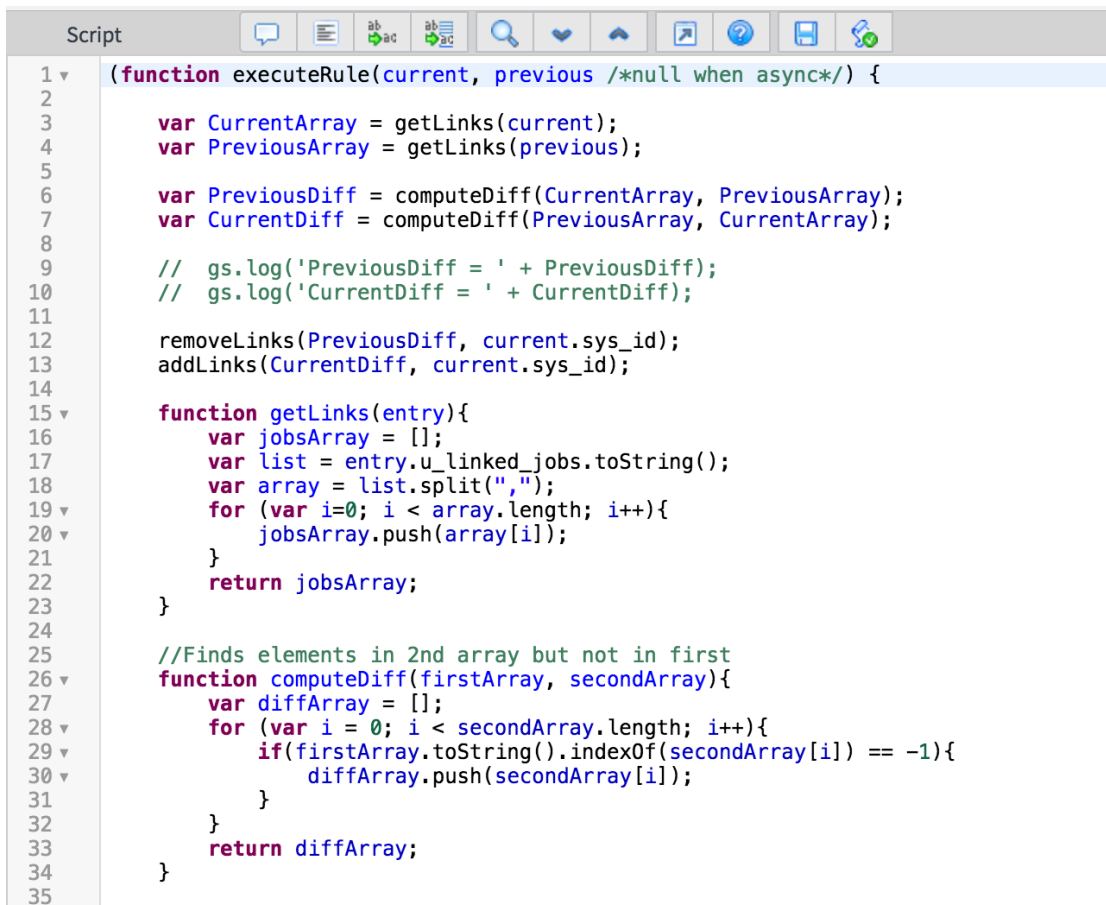
Figure 61. Calculate SD metrics from job.

This rule is triggered every time an active job is created or updated.

It calls the Script Include 'AIM_getMetricBilling' and its function 'getSDBillingMetrics' which calculates the metrics for the current job as dictated by

various conditions. We'll look at the Script include in the Script Include section of this paper.

8) Update Linked Jobs



```

1 (function executeRule(current, previous /*null when async*/) {
2
3     var CurrentArray = getLinks(current);
4     var PreviousArray = getLinks(previous);
5
6     var PreviousDiff = computeDiff(CurrentArray, PreviousArray);
7     var CurrentDiff = computeDiff(PreviousArray, CurrentArray);
8
9     // gs.log('PreviousDiff = ' + PreviousDiff);
10    // gs.log('CurrentDiff = ' + CurrentDiff);
11
12    removeLinks(PreviousDiff, current.sys_id);
13    addLinks(CurrentDiff, current.sys_id);
14
15    function getLinks(entry){
16        var jobsArray = [];
17        var list = entry.u_linked_jobs.toString();
18        var array = list.split(",");
19        for (var i=0; i < array.length; i++){
20            jobsArray.push(array[i]);
21        }
22        return jobsArray;
23    }
24
25    //Finds elements in 2nd array but not in first
26    function computeDiff(firstArray, secondArray){
27        var diffArray = [];
28        for (var i = 0; i < secondArray.length; i++){
29            if(firstArray.toString().indexOf(secondArray[i]) == -1){
30                diffArray.push(secondArray[i]);
31            }
32        }
33        return diffArray;
34    }
35

```

Figure 62: Update linked jobs.

This rule is triggered every time an active job is inserted or updated.

The linked jobs field is a list type field, so every time the field is updated, we need to make sure that the new data is appended to the existing values, and not replacing it.

```

Script
diffArray.push(secondArray[i]);
    }
}
return diffArray;
}

function removeLinks(PreviousDiff, sysID){
    // gs.log("Running removeLinks - " + PreviousDiff);
    for (var i = 0; i < PreviousDiff.length; i++){
        var gr = new GlideRecord('u_service_delivery_job');
        // gs.log("This is the PreviousDiff[i] - " + PreviousDiff[i]);
        gr.addQuery('sys_id',PreviousDiff[i]);
        gr.query();
        while(gr.next()){
            var list = gr.u_linked_jobs.toString();
            if(list.indexOf(sysID) >= 0){
                list = list.replace(sysID.toString(),"");
                // gs.log("This is being removed - " + sysID);
                gr.u_linked_jobs = list;
                gr.updateWithReferences();
                // gs.log("Removed Links New List - " + list);
            }
            else{
                // gs.log("Nothing to remove");
            }
        }
    }
}

function addLinks(CurrentDiff, sysID){
    // gs.log("Running addLinks - " + CurrentDiff);
    for (var i = 0; i < CurrentDiff.length; i++){
        var gr = new GlideRecord('u_service_delivery_job');
        gr.addQuery('sys_id',CurrentDiff[i]);
        gr.query();
        while(gr.next()){
            //gs.log("In While AddLinks - ")
            if(gr.u_linked_jobs != ''){
                var newLink = gr.u_linked_jobs.toString();
                var concatLink = newLink.concat(", " + sysID);
                //gs.log("newLink = " + gr.u_linked_jobs.toString());
                //gs.log("New newLink created - " + newLink);
                //gs.log("New concatLink created - " + concatLink);
                //gs.log("Adding to list - " + sysID);
                gr.u_linked_jobs = concatLink;
                gr.updateWithReferences();
            }
            else{
                gr.u_linked_jobs = sysID;
                gr.updateWithReferences();
            }
        }
    }
}
})(current, previous);

```

Figure 63. Update linked jobs.

9) Evidence Update

The screenshot shows a configuration form for a business rule named 'Evidence Update'. The 'Name' field is 'Evidence Update', the 'Table' is 'Service Delivery Job [u_service_delivery_j...', the 'Application' is 'Global', and both 'Active' and 'Advanced' checkboxes are checked. Below the form, the 'Script' field contains the following code:

```

1 (function executeRule(current, previous /*null when async*/) {
2
3   // Add your code here
4
5   var lhe = new ServiceDeliveryUtil();
6
7   current.u_evidence = lhe.evidenceStringUpdate(current.u_evidence_number.toString());
8
9
10  })(current, previous);
11

```

Figure 64. Evidence update.

Evidence is another list field that stores the references to the Evidence records.

This rule is triggered every time the 'Evidence Number' field on the Job record is updated.

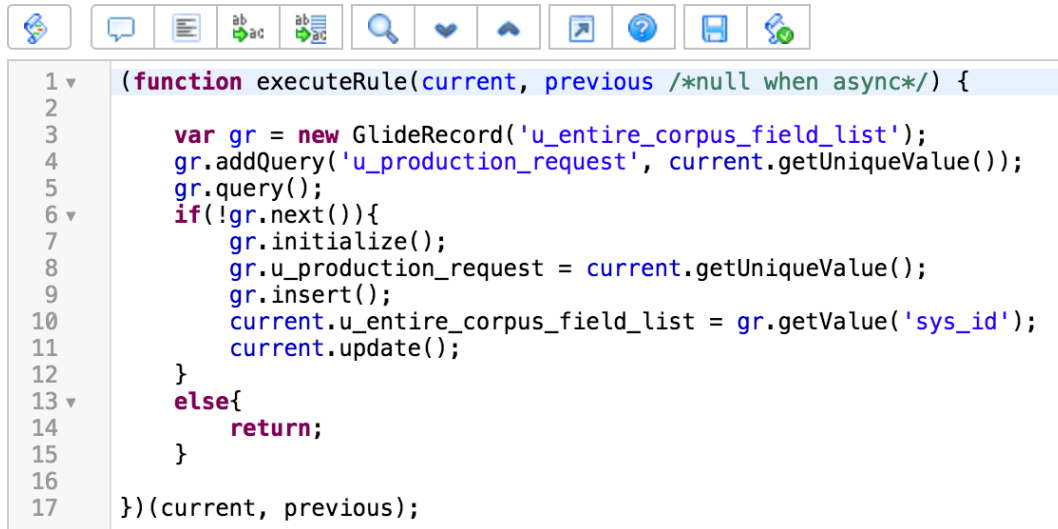
It calls the Script Include 'ServiceDeliveryUtil' and its method 'evidenceStringUpdate()'

We'll look at the Script Include in the Script Include section of this paper.

Next up are the Business Rules implemented for 'Production Request' table.

10) Create Entire Corpus Field List

This rule is to ensure there's one Entire Corpus field list record associated with every Production Request record and create one if one doesn't exist.



```

1  (function executeRule(current, previous /*null when async*/) {
2
3      var gr = new GlideRecord('u_entire_corpus_field_list');
4      gr.addQuery('u_production_request', current.getUniqueValue());
5      gr.query();
6  if(!gr.next()){
7      gr.initialize();
8      gr.u_production_request = current.getUniqueValue();
9      gr.insert();
10     current.u_entire_corpus_field_list = gr.getValue('sys_id');
11     current.update();
12 }
13 else{
14     return;
15 }
16
17 })(current, previous);

```

Figure 65. Create entire corpus field list.

11) EC Fields List

This rule is triggered every time a Production Request's state is changed to Ready.

The objective of this script is to gather all the unique 'Source Fields' referenced in various fields on the current request, its associated records and their associated records and save the list in the Entire Corpus field (created in the above rule) referenced in the current record.

To achieve this, this script queries the following tables: Endorsements, Builds, Load Files, Load File Source Fields, Field Condition Builders, Concatenations, Placeholder Specifications, and Identifications.

```

Script
1 (function executeRule(current, previous /*null when async*/) {
2
3     var fields = [];
4
5     //Source fields from Builds
6     //gs.log('veena builds ' + current.u_builds);
7     var gr = new GlideRecord('u_build');
8     gr.addQuery('sys_id', 'IN', current.u_builds);
9     gr.query();
10    while(gr.next()){
11        //gs.log('veena build: ' + gr.u_name);
12        var gr2 = new GlideRecord('u_load_file');
13        gr2.addQuery('sys_id', 'IN', gr.u_load_file_reference);
14        gr2.query();
15        while(gr2.next()){
16            //gs.log('veena load files ' + gr.u_load_file_reference);
17            var gr3 = new GlideRecord('u_load_file_source_list');
18            gr3.addQuery('u_load_file', gr2.getUniqueValue());
19            gr3.addQuery('u_source_field_name.u_source', 'Relativity');
20            gr3.query();
21            while(gr3.next()){
22                //gs.log('veena lfsf: ' + gr3.u_label);
23                fields.push(gr3.u_source_field_name.u_field_name.toString());
24                //fields.push('\n');
25            }
26        }
27    }
28    /*for (var i=0; i < fields.length; i++) {
29        gs.log('veena field array from builds: ' + fields[i]);
30    }*/
31    //Source fields from Endorsements
32    var gre = new GlideRecord('u_endorsements');
33    gre.addQuery('sys_id', 'IN', current.u_endorsement_s);
34    gre.addQuery('u_source_field.u_source', 'Relativity');
35    gre.query();
36    while(gre.next()){
37        //gs.log('veena endorsement: ' + gre.u_endorsement);
38        fields.push(gre.u_source_field.u_field_name.toString());
39        //fields.push('\n');
40    }
41
42    //Source fields from PR 'Sort fields' list field.
43    var grs = new GlideRecord('u_source_field_list');
44    grs.addQuery('sys_id', 'IN', current.u_sort_field);
45    grs.addQuery('u_source', 'Relativity');
46    grs.query();
47    while(grs.next()){
48        fields.push(grs.u_field_name.toString());
49        //fields.push('\n');
50    }
51

```

Figure 66. EC fields list.

```

50 }
51
52 //Source fields from Placeholder Specifications Identifications
53 var grp = new GlideRecord('u_specs');
54 grp.addQuery('sys_id', 'IN', current.u_placeholder_specifications);
55 grp.query();
56 while(grp.next()){
57     var gri = new GlideRecord('u_identifications');
58     gri.addQuery('u_placeholder_specification', grp.getUniqueValue());
59     gri.addQuery('u_first_operand.u_source', 'Relativity');
60     gri.query();
61     while(gri.next()){
62         fields.push(gri.u_first_operand.u_field_name.toString());
63         //fields.push('\n');
64     }
65 }
66
67 //Source fields with 'Auto include on Entire Corpus Field List' set to true
68 var grsf = new GlideRecord('u_source_field_list');
69 grsf.addQuery('u_auto_include_on_entire_corpus_field_list', 'true');
70 //grsf.addQuery('u_source', 'Relativity');
71 grsf.query();
72 while(grsf.next()){
73     fields.push(grsf.u_field_name.toString());
74     //fields.push('\n');
75 }
76
77 //Source Fields in Field Condition Builders/Concatenations coming from Builds
78 var grb = new GlideRecord('u_build');
79 grb.addQuery('sys_id', 'IN', current.u_builds);
80 grb.query();
81 //gs.log('veena builds: ' + current.u_builds);
82 while(grb.next()){
83     var grlf = new GlideRecord('u_load_file');
84     grlf.addQuery('sys_id', 'IN', grb.u_load_file_reference);
85     grlf.query();
86     //gs.log('veena load files: ' + grb.u_load_file_reference);
87     while(grlf.next()){
88         var grlfs = new GlideRecord('u_load_file_source_list');
89         grlfs.addQuery('u_load_file', grlf.getUniqueValue());
90         grlfs.query();
91         while(grlfs.next()){
92             //gs.log('veena lfsf: ' + grlfs.u_label);
93             if(grlfs.u_additional_field_options == 'Field Condition Builder'){
94                 //gs.log('veena: in fcb');
95                 var grfcb = new GlideRecord('u_field_condition_builder');
96                 grfcb.addQuery('u_load_file_source_field', grlfs.getUniqueValue());
97                 //grfcb.addQuery('u_first_operand_source_field.u_source', 'Relativity');
98                 grfcb.query();

```

Figure 67. EC fields list.

```

98         grfcb.query();
99         while(grfcb.next()){
100             if(grfcb.u_first_operand_source_field.u_source == 'Relativity'){
101                 //gs.log('veena fcb: ' + grfcb.u_first_operand_source_field.u_field_name);
102                 fields.push(grfcb.u_first_operand_source_field.u_field_name.toString());
103             }
104             if(grfcb.u_source_field_value.u_source == 'Relativity'){
105                 //gs.log('veena fcb: ' + grfcb.u_source_field_value.u_field_name);
106                 fields.push(grfcb.u_source_field_value.u_field_name.toString());
107             }
108         }
109     }
110     else if(grlfs.u_additional_field_options == 'Concatenation'){
111         var grc = new GlideRecord('u_concatenation');
112         grc.addQuery('u_load_file_source_field', grlfs.getUniqueValue());
113         grc.addQuery('u_source_field.u_source', 'Relativity');
114         grc.query();
115         while(grc.next()){
116             //gs.log('veena concat: ' + grc.u_source_field.u_field_name);
117             fields.push(grc.u_source_field.u_field_name.toString());
118         }
119     }
120 }
121 }
122 }
123
124 fields.sort();
125 fields = new ArrayUtil().unique(fields);
126 fields = fields.toString();
127 fields = fields.replace(/,/g, '\n');
128 //fields.split(",").join("\n");
129
130 //gs.log('veena ec list: ' + fields);
131 var ec = new GlideRecord('u_entire_corpus_field_list');
132 ec.addQuery('u_production_request', current.sys_id);
133 ec.query();
134 while(ec.next()){
135     ec.u_entire_corpus_field_list = fields;
136     ec.update();
137 }
138
139 })(current, previous);

```

Figure 68. EC fields list

12) Set FPNI Query on PR

This script is triggered every time a PR job record is inserted or updated. It queries all the FPNs referenced in the current record and forms a query from each of those records.

```

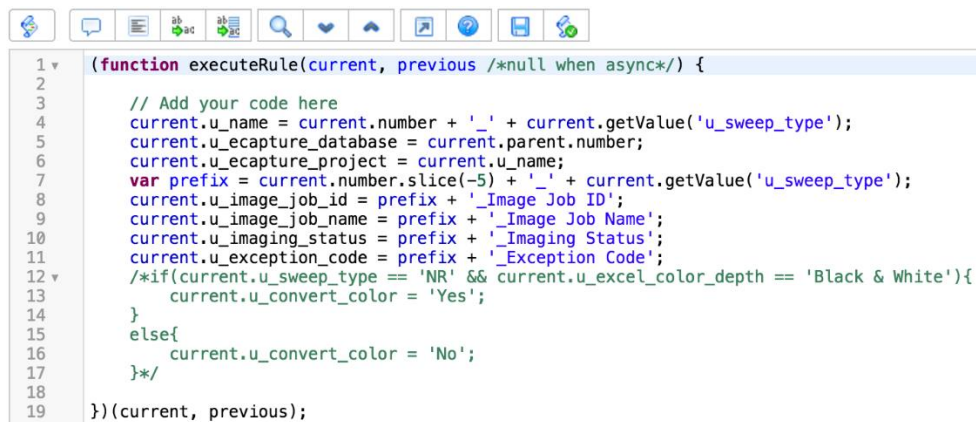
1  (function executeRule(current, previous /*null when async*/) {
2
3     var query = '';
4     //gs.log('Current FPNs: ' + current.u_fpn_imaged);
5     var list = current.u_fpn_imaged.toString();
6     var fpn = list.split(",");
7     //gs.log('FPN Length is ' + fpn.length + '\n');
8     for(var i = 0; i < fpn.length; i++){
9         //gs.log("Reference value is: " + fpn[i]);
10        var gr = new GlideRecord('u_file_provided_natively_imaged');
11        gr.addQuery('sys_id', fpn[i]);
12        gr.query();
13        if(gr.next()){
14            //gs.log('FPN Imaged is: ' + gr.u_name + '\n');
15            query += gr.getValue('u_first_operand') + ' ' + gr.getValue('u_operator') + ' ' +
16                gr.getValue('u_second_operand') + '\n';
17        }
18    }
19    query = query.replace(/null/g, '');
20    //gs.log('Veena FPNI Test: ' + query + '\n');
21    current.u_query = query;
22
23 })(current, previous);

```

Figure 69. Set FPNI query on PR.

Next, we will be looking at the Business Rules created for the Image Sweep request table.

13) Image Sweep AutoFill fields



```

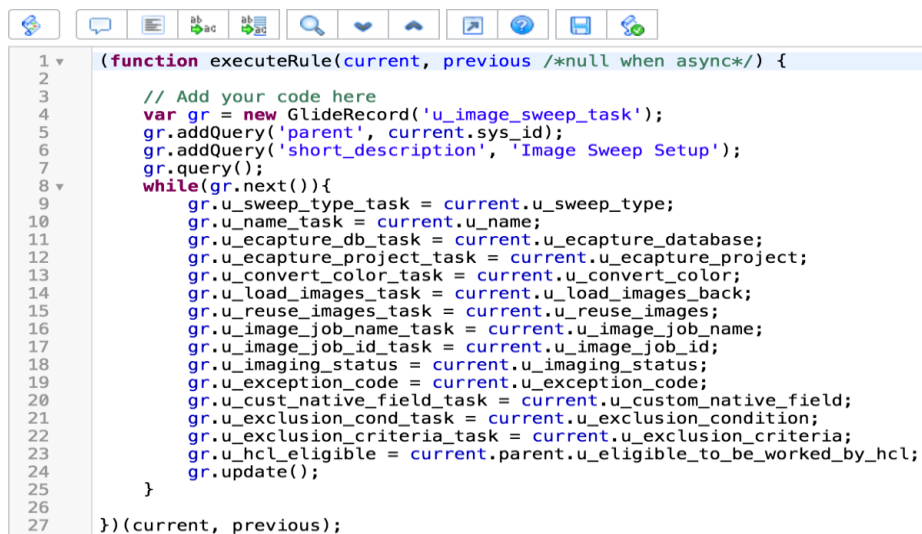
1  (function executeRule(current, previous /*null when async*/) {
2
3      // Add your code here
4      current.u_name = current.number + '_' + current.getValue('u_sweep_type');
5      current.u_ecapture_database = current.parent.number;
6      current.u_ecapture_project = current.u_name;
7      var prefix = current.number.slice(-5) + '_' + current.getValue('u_sweep_type');
8      current.u_image_job_id = prefix + '_Image Job ID';
9      current.u_image_job_name = prefix + '_Image Job Name';
10     current.u_imaging_status = prefix + '_Imaging Status';
11     current.u_exception_code = prefix + '_Exception Code';
12     /*if(current.u_sweep_type == 'NR' && current.u_excel_color_depth == 'Black & White'){
13         current.u_convert_color = 'Yes';
14     }
15     else{
16         current.u_convert_color = 'No';
17     }*/
18
19 })(current, previous);

```

Figure 70. Image sweep autoFill fields.

This rule is triggered before an Image Sweep Record is inserted or updated to ensure certain fields on the record are always consistent and based on Sweep Type and the FQ Job code.

14) Sync IS Task fields



```

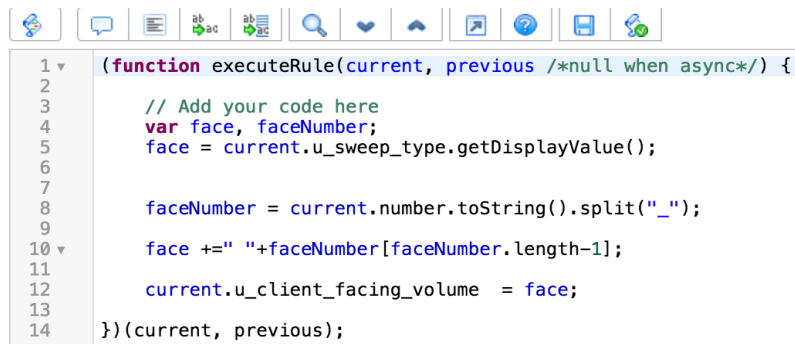
1  (function executeRule(current, previous /*null when async*/) {
2
3      // Add your code here
4      var gr = new GlideRecord('u_image_sweep_task');
5      gr.addQuery('parent', current.sys_id);
6      gr.addQuery('short_description', 'Image Sweep Setup');
7      gr.query();
8      while(gr.next()){
9          gr.u_sweep_type_task = current.u_sweep_type;
10         gr.u_name_task = current.u_name;
11         gr.u_ecapture_db_task = current.u_ecapture_database;
12         gr.u_ecapture_project_task = current.u_ecapture_project;
13         gr.u_convert_color_task = current.u_convert_color;
14         gr.u_load_images_task = current.u_load_images_back;
15         gr.u_reuse_images_task = current.u_reuse_images;
16         gr.u_image_job_name_task = current.u_image_job_name;
17         gr.u_image_job_id_task = current.u_image_job_id;
18         gr.u_imaging_status = current.u_imaging_status;
19         gr.u_exception_code = current.u_exception_code;
20         gr.u_cust_native_field_task = current.u_custom_native_field;
21         gr.u_exclusion_cond_task = current.u_exclusion_condition;
22         gr.u_exclusion_criteria_task = current.u_exclusion_criteria;
23         gr.u_hcl_eligible = current.parent.u_eligible_to_be_worked_by_hcl;
24         gr.update();
25     }
26
27 })(current, previous);

```

Figure 71. Sync IS task fields.

This script ensures that the fields on the Image sweep task are always consistent with the fields on the associated Image Sweep Request.

15) Generate Client Facing Volume



```

1 (function executeRule(current, previous /*null when async*/) {
2
3 // Add your code here
4 var face, faceNumber;
5 face = current.u_sweep_type.getDisplayValue();
6
7
8 faceNumber = current.number.toString().split("_");
9
10 face += " "+faceNumber[faceNumber.length-1];
11
12 current.u_client_facing_volume = face;
13
14 })(current, previous);

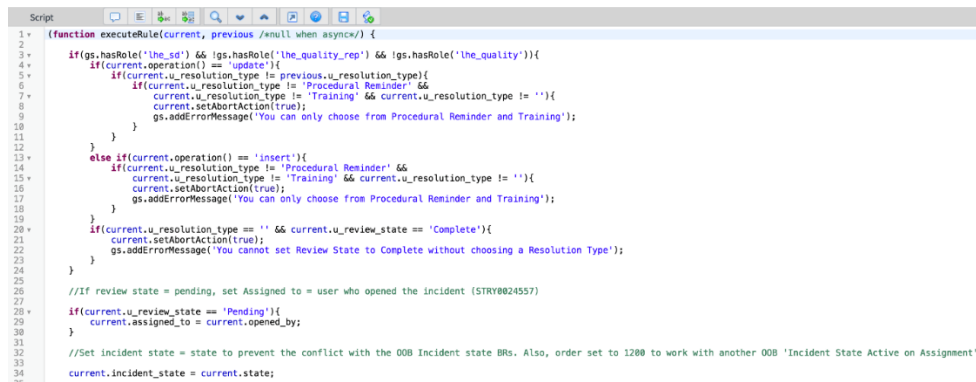
```

Figure 72. Generate client facing volume

This script is to generate the 'Client Facing Volume' that is utilized by SL.

The next couple of Business Rules run on 'Quality Incident' table extended from the Out of Box Incident table.

16) Check Validity before Submit



```

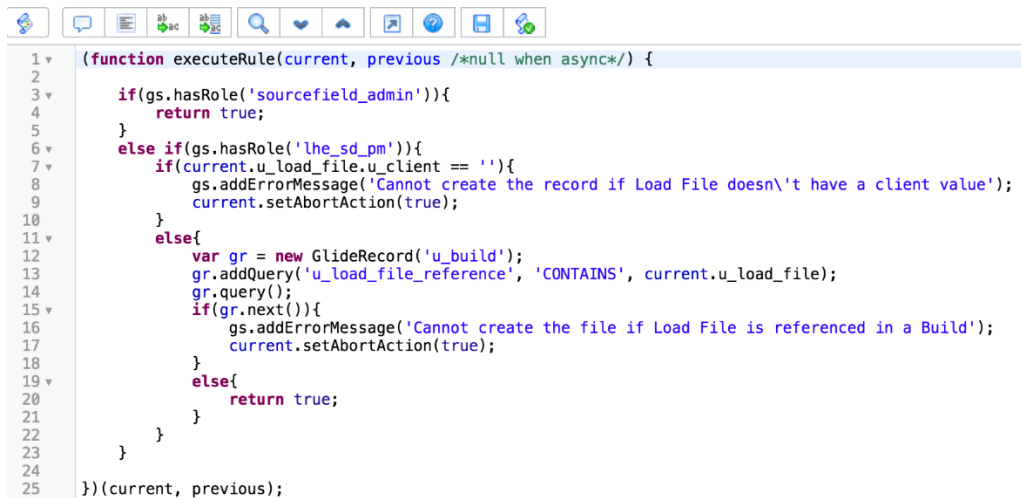
1 (function executeRule(current, previous /*null when async*/) {
2
3 if(gs.hasRole('the_sd') && !gs.hasRole('the_quality_rep') && !gs.hasRole('the_quality')){
4 if(current.operation() == 'update'){
5 if(current.u_resolution_type != previous.u_resolution_type){
6 if(current.u_resolution_type == 'Procedural Reminder' &&
7 current.u_resolution_type != 'Training' && current.u_resolution_type != ''){
8 current.setAbortAction(true);
9 gs.addErrorMessage('You can only choose from Procedural Reminder and Training');
10 }
11 }
12 }
13 } else if(current.operation() == 'insert'){
14 if(current.u_resolution_type != 'Procedural Reminder' &&
15 current.u_resolution_type != 'Training' && current.u_resolution_type != ''){
16 current.setAbortAction(true);
17 gs.addErrorMessage('You can only choose from Procedural Reminder and Training');
18 }
19 }
20 }
21 if(current.u_resolution_type == '' && current.u_review_state == 'Complete'){
22 current.setAbortAction(true);
23 gs.addErrorMessage('You cannot set Review State to Complete without choosing a Resolution Type');
24 }
25 }
26 //If review state = pending, set Assigned to = user who opened the incident (STRY0024557)
27
28 if(current.u_review_state == 'Pending'){
29 current.assigned_to = current.opened_by;
30 }
31
32 //Set incident state = state to prevent the conflict with the 00B Incident state BRs. Also, order set to 1200 to work with another 00B 'Incident State Active on Assignment'
33
34 current.incident_state = current.state;
35
36 }

```

Figure 73. Check validity before submit.

The next rule is implemented on Load File Source Fields table, but there are other rules implemented on other tables but they bear the same functionality, hence only one is described here.

17) Check conditions before submitting



```

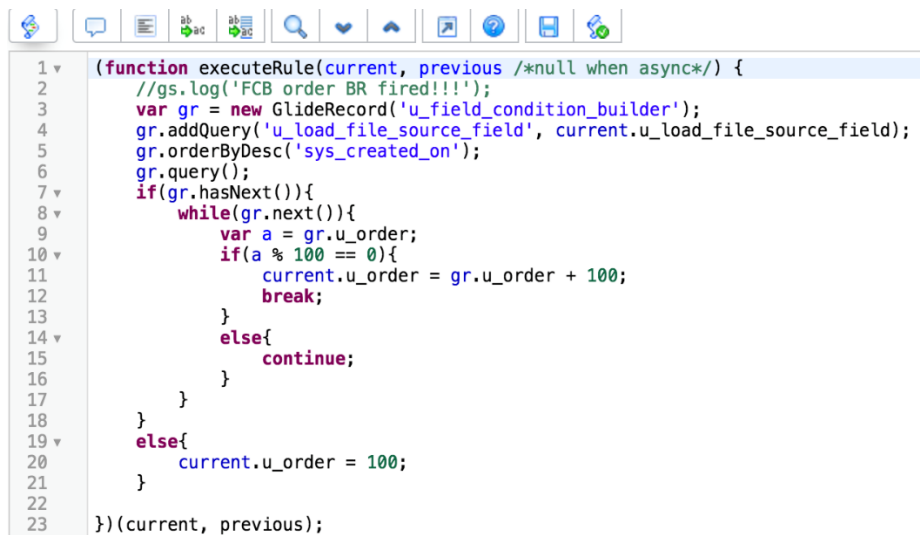
1  (function executeRule(current, previous /*null when async*/) {
2
3  if(gs.hasRole('sourcefield_admin')){
4      return true;
5  }
6  else if(gs.hasRole('lhe_sd_pm')){
7      if(current.u_load_file.u_client == ''){
8          gs.addErrorMessage('Cannot create the record if Load File doesn't have a client value');
9          current.setAbortAction(true);
10     }
11     else{
12         var gr = new GlideRecord('u_build');
13         gr.addQuery('u_load_file_reference', 'CONTAINS', current.u_load_file);
14         gr.query();
15         if(gr.next()){
16             gs.addErrorMessage('Cannot create the file if Load File is referenced in a Build');
17             current.setAbortAction(true);
18         }
19         else{
20             return true;
21         }
22     }
23 }
24
25 })(current, previous);

```

Figure 74. Check conditions before submitting.

18) Set Order on Field Cond Builder

This rule is implemented on Field Condition Builder table to set the Order field in increments of 100 for every set of records associated with a single load file source field record.



```

1  (function executeRule(current, previous /*null when async*/) {
2      //gs.log('FCB order BR fired!!!');
3      var gr = new GlideRecord('u_field_condition_builder');
4      gr.addQuery('u_load_file_source_field', current.u_load_file_source_field);
5      gr.orderByDesc('sys_created_on');
6      gr.query();
7      if(gr.hasNext()){
8          while(gr.next()){
9              var a = gr.u_order;
10             if(a % 100 == 0){
11                 current.u_order = gr.u_order + 100;
12                 break;
13             }
14             else{
15                 continue;
16             }
17         }
18     }
19     else{
20         current.u_order = 100;
21     }
22 }
23 })(current, previous);

```

Figure 75. Set order on field cond builder.

19) Set Load File Field Labels

This rule is created for Load File Source Fields table which collects all the source fields associated with a single load file and stores those field names in a string field on the associated Load File record.

```

1 (function executeRule(current, previous /*null when async*/) {
2
3     var labels = '', gr = new GlideRecord('u_load_file_source_list');
4     gr.addQuery('u_load_file', current.u_load_file);
5     gr.orderBy('u_order');
6     gr.query();
7     while(gr.next()){
8         labels += gr.u_label + '\n';
9     }
10    //gs.log('Labels for ' + current.u_load_file.u_name + ' are: ' + labels);
11    var lf = new GlideRecord('u_load_file');
12    lf.addQuery('sys_id', current.u_load_file);
13    lf.query();
14    while(lf.next()){
15        lf.u_load_file_field_labels = labels;
16        lf.update();
17    }
18
19 })(current, previous);

```

Figure 76. Set load file field labels.

20) Capture Query for Exception

```

Script
1 (function executeRule(current, previous /*null when async*/) {
2
3     //gs.log('current type: ' + current.u_spec_type + ' and previous type: ' + previous.u_spec_type);
4     if(current.u_spec_type == 'excep' && previous.u_spec_type != 'excep'){
5
6         var gr = new GlideRecord('u_identifications');
7         gr.addQuery('u_placeholder_specification', current.getUniqueValue());
8         gr.query();
9         gr.deleteMultiple();
10
11        var grr = new GlideRecord('u_identifications');
12        grr.initialize();
13        grr.u_placeholder_specification = current.getUniqueValue();
14        grr.u_first_operand = '40659cbd1389e600bcf1d6e2e144b02d';
15        grr.u_operator = '=';
16        grr.u_second_operand = '1';
17        grr.u_condition = '1';
18        grr.insert();
19
20        var query = '';
21        var grid = new GlideRecord('u_identifications');
22        grid.addQuery('u_placeholder_specification', current.getUniqueValue());
23        grid.query();
24        while(grid.next()){
25            //gs.log("Field and Source: " + grid.u_first_operand.u_field_name + " and " + grid.u_first_operand.u_source);
26            if(grid.getValue('u_condition')){
27                query += grid.u_first_operand.u_source + " " + grid.u_first_operand.u_field_name + " " + grid.getValue('u_operator') + " " +
28                grid.getValue('u_second_operand') + "\n" + grid.getValue('u_condition') + "\n";
29            }
30            else{
31                query += grid.u_first_operand.u_source + " " + grid.u_first_operand.u_field_name + " " + grid.getValue('u_operator') + " " +
32                grid.getValue('u_second_operand') + "\n";
33            }
34        }
35        current.u_query = query;
36    }
37 }

```

Figure 77. Capture query for exception.

This runs on the Placeholder Specifications table when the Specifications type changes to Exception.

Script includes. The following are the custom Script Includes created to facilitate reusable code both from Business Rules and Client Scripts.

1) AIM_getMetricBilling

This is the Script Include referenced in one of the Business Rules above.

```

Script
1  var AIM_getMetricBilling = Class.create();
2  AIM_getMetricBilling.prototype = {
3  initialize: function() {
4  },
5  //Initial Function that runs to collect the Metrics associated with the current Service Delivery Job
6  getSDBillingMetrics: function(sysId, billItems, jStatus, parent, jobType){
7      var metrics, sdJob, billItem, dataPoint, matter, rate, metric, srate, billArray, exception;
8      metrics = new GlideRecord('metric_instance');
9      metrics.addQuery('id', sysId);
10     metrics.query();
11     while(metrics.next()){
12         //Loop through the metrics and do these actions to each metric
13         billItem = billItems.split(',');
14         dataPoint = metrics.u_imported_field.toString();
15         matter = parent.sys_id;
16         exception = metrics.u_exception;
17         //gs.log('current data point - ' + dataPoint);
18
19         //Populate Rate for the current Data Point//
20         //Special Data Points with special rules
21
22         //If Data Point = "OCR Page Count"
23         if (dataPoint == "OCR Page Count"){
24             //gs.log('This is an OCR Page Count - ' + metrics.id.toString());
25             metrics.u_rate = this.getBillRate(parent,'PROC.OCR');
26             metrics.u_billable_item_code = "PROC.OCR";
27             metrics.u_calculated_charge = metrics.u_rate*metrics.u_value;
28             metrics.update();
29         }
30         else if (dataPoint == "Hard Drives (over 250GB)") {
31             //gs.log('This is an MEDIA.LG.HD - ' + metrics.id.toString());
32             metrics.u_rate = this.getBillRate(parent,'MEDIA.LG.HD');
33             metrics.u_billable_item_code = "MEDIA.LG.HD";
34             metrics.u_calculated_charge = metrics.u_rate*metrics.u_value;
35             metrics.update();
36         }
37         else if (dataPoint == "DVDs" && exception != true){
38             //gs.log('This is a DVD - ' + metrics.id.toString());
39             metrics.u_rate = this.getBillRate(parent,'MEDIA.DVD');
40             metrics.u_billable_item_code = "MEDIA.DVD";
41             metrics.u_calculated_charge = metrics.u_rate*metrics.u_value;
42             metrics.update();
43         }
44         else if (dataPoint == "Client Archive Request"){
45             //gs.log("This is a Client Archive Request");
46             metrics.u_rate = this.getBillRate(parent,"HOST.ARCHIVE.FEE");
47             metrics.u_billable_item_code = "HOST.ARCHIVE.FEE";
48             metrics.u_calculated_charge = metrics.u_rate*metrics.u_value;
49             metrics.update();
50         }

```

Figure 78. AIM_getMetricBilling.

```

51 *     else if ((dataPoint == "Ingested GB") && (metrics.u_value < 2) && (jobType == "ESI Review Volume") && this.getBillRate(parent, 'PROC.MINIMUM') != 0){
52 *         metrics.u_rate = this.getBillRate(parent, 'PROC.MINIMUM');
53 *         //gs.log("Calculating PROC.MINIMUM");
54 *         metrics.u_billable_item_code = "PROC.MINIMUM";
55 *         metrics.u_calculated_charge = metrics.u_rate;
56 *         metrics.update();
57 *     }
58 *     else if (exception == true){
59 *         //gs.log("Running exception...");
60 *         for (var j = 0; j < billItem.length; j++){
61 *             //gs.log("Before Normalize - " + billItem[j]);
62 *             billItem[j] = this.normalizeBillItem(billItem[j]);
63 *             //gs.log("After Normalize - " + billItem[j]);
64 *             rate = this.getJobMetricChargeCalc(dataPoint, billItem[j], matter, exception.toString());
65 *             if (rate == 0){
66 *                 //gs.log("Clearing out Rate for - " + billItem[j] + " - " + metrics.u_imported_field);
67 *                 metrics.u_rate = '';
68 *                 metrics.u_billable_item_code = '';
69 *                 metrics.u_calculated_charge = '';
70 *                 metrics.update();
71 *             }
72 *             else{
73 *                 //gs.log('Rate should be - ' + rate + ' for Bill Item = ' + billItem[j] + " - " + metrics.u_imported_field);
74 *                 metrics.u_rate = rate;
75 *                 metrics.u_billable_item_code = billItem[j];
76 *                 metrics.u_calculated_charge = metrics.u_rate*metrics.u_value;
77 *                 metrics.update();
78 *                 break;
79 *             }
80 *         }
81 *     }
82 *     else{
83 *         billArray = [];
84 *         //Loop through the Bill items on the SD Job
85 *         for (var i = 0; i < billItem.length; i++){
86 *             billItem[i] = this.normalizeBillItem(billItem[i]);
87 *             //gs.log("Current Bill Item in Loop - " + billItem[i] + ' with dataPoint - ' + dataPoint);
88 *             var item = billItem[i];
89 *             billArray.push(item.toString());
90 *             rate = this.getJobMetricChargeCalc(dataPoint, billItem[i], matter, exception);
91 *             if (rate == 0){
92 *                 //gs.log("Clearing out Rate for - " + billItem[i] + " - " + metrics.u_imported_field);
93 *                 metrics.u_rate = '';
94 *                 metrics.u_billable_item_code = '';
95 *                 metrics.u_calculated_charge = '';
96 *                 metrics.update();
97 *             }
98 *             else{
99 *                 //gs.log('Rate should be - ' + rate + ' for Bill Item = ' + billItem[i] + " - " + metrics.u_imported_field);
100 *                 metrics.u_rate = rate;
101 *                 metrics.u_billable_item_code = billItem[i];
102 *                 metrics.u_calculated_charge = metrics.u_rate*metrics.u_value;
103 *                 metrics.update();
104 *                 break;

```

Figure 79. AIM_getMetricBilling.

```

108 *         //Special Rule for PROD.MINIMUM
109 *         var splitArray = billArray.toString();
110 *         //gs.log("Items in the splitArray = " + splitArray + ' Job Type is - ' + jobType + ' Metrics Value = ' + metrics.u_value);
111 *         if (jobType == "u_production_request" && metrics.u_value < 2 && splitArray.indexOf("PROD.BRAND") > -1 && splitArray.indexOf("PROD.IMAGE") > -1){
112 *             //gs.log("Calculating the Minimum");
113 *             srate = this.getJobMetricChargeCalc(dataPoint, "PROD.MINIMUM", matter, exception);
114 *             if (srate == 0){
115 *                 //gs.log("No PROD.MINIMUM");
116 *                 continue;
117 *             }
118 *             else{
119 *                 //gs.log("Found PROD.MINIMUM!!!!");
120 *                 metrics.u_rate = srate;
121 *                 metrics.u_billable_item_code = "PROD.MINIMUM";
122 *                 metrics.u_calculated_charge = metrics.u_rate;
123 *                 metrics.update();
124 *             }
125 *         }
126 *     }
127 * },
128 * },
129 *
130 * getJobMetricChargeCalc: function(dataPoint, billItem, matter, exception){
131 *     var calc, uom;
132 *     //gs.log("Running getJobMetricChargeCalc with values - " + dataPoint + " , " + billItem + " , " + matter + " , " + exception);
133 *     calc = new GlideRecord('u_job_metrics_charge_calculations');
134 *     calc.addQuery('u_data_point', dataPoint.toString());
135 *     calc.addQuery('u_billing_item_code', billItem.toString());
136 *     calc.addQuery('u_exception', exception);
137 *     calc.addQuery('u_active', true);
138 *     calc.query();
139 *
140 *     if(!calc.hasNext()){
141 *         //gs.log("No match of ' + billItem + ' - ' + dataPoint);
142 *         return 0;
143 *     }
144 *     while(calc.next()){
145 *         uom = calc.u_unit_of_measure;
146 *         //gs.log("getJobMetricChargeCalc returning values - ' + billItem + ', ' + uom + ', ' + matter);
147 *         return this.getBillingRate(billItem, uom, matter);
148 *     }
149 * },
150 *
151 * getBillRate: function(matter, billItem){
152 *     var billRate = new GlideRecord('u_matter_billing_rates');
153 *     //gs.log("Bill Rate Calc - " + billItem);
154 *     billRate.addQuery('u_matter', matter);
155 *     billRate.addQuery('u_bill_code', billItem);
156 *     billRate.addQuery('u_active', true);
157 *     billRate.query();
158 *     if(!billRate.hasNext()){
159 *         return 0;
160 *     }
161 *     while(billRate.next()){
162 *         //gs.log("Bill Rate Value - " + billRate.u_rate);
163 *         return billRate.u_rate;
164 *     }

```

Figure 80. AIM_getMetricBilling.

```

167 ▾   getBillingRate: function(billItem,uom,matter){
168       var rate;
169       rate = new GlideRecord('u_matter_billing_rates');
170       rate.addQuery('u_matter',matter);
171       rate.addQuery('u_bill_code',billItem);
172       rate.addQuery('u_measure',uom);
173       rate.addQuery('u_active','true');
174       //gs.log('getBillingRate Running - ' + billItem + ' - ' + uom + ' - ');
175       rate.query();
176
177 ▾       if(!rate.hasNext()){
178           return 0;
179       }
180
181
182 ▾       while(rate.next()){
183           return rate.u_rate;
184       }
185     },
186
187 ▾   normalizeBillItem: function(billItem){
188 ▾       switch (billItem.toString()){
189
190           case 'PROD-BRAND':
191             if(billItem.indexOf('PROD-BRAND') != -1)
192               return 'PROD.BRAND';
193             break;
194           case 'PROD-IMG2IMG':
195             if(billItem.indexOf('PROD-IMG2IMG') != -1)
196               return 'PROD.IMG2IMG';
197             break;
198           case 'PROD-IMAGE':
199             if(billItem.indexOf('PROD-IMAGE') != -1)
200               return 'PROD.IMAGE';
201             break;
202           case 'PROC-EXPORT':
203             if(billItem.indexOf('PROC-EXPORT') != -1)
204               return 'PROC.EXPORT';
205             break;
206           case 'PROC-HYBRID':
207             if(billItem.indexOf('PROC-HYBRID') != -1)
208               return 'PROC.HYBRID';
209             break;
210           case 'PROC-OCR':
211             if(billItem.indexOf('PROC-OCR') != -1)
212               return 'PROC.OCR';
213             break;
214           case 'PROC-INGEST':
215             if(billItem.indexOf('PROC-INGEST') != -1)
216               return 'PROC.INGEST';
217             break;
218           case 'PROC-FILTER':
219             if(billItem.indexOf('PROC-FILTER') != -1)
220               return 'PROC.FILTER';
221             break;
222           case 'PROC-PREPROC':

```

Figure 81. AIM_getMetricBilling.

```

221         break;
222     case 'PROC-PREPROC':
223         if(billItem.indexOf('PROC-PREPROC') != -1)
224             return 'PROC.PREPROC';
225     },
226
227     autoCreateMetric: function(){
228         //gs.log('JLM - Starting autoCreateMetric');
229         var gr = new GlideRecord('u_matter_billing_rates');
230         gr.addEncodedQuery('u_matter.state=2^u_active=true^u_bill_codeLIKEPROD.IMAGE^u_measureLIKEPAGE');
231         gr.query();
232         while(gr.next()){
233             //gs.log('JLM - Querying Matter Code - ' + gr.u_matter);
234             var gr1 = new GlideRecord('u_service_delivery_job');
235             gr1.addQuery('parent', gr.u_matter);
236             gr1.addQuery('active', 'true');
237             gr1.addQuery('u_billable_component', 'CONTAINS', 'PROD-IMAGE');
238             gr1.query();
239             while(gr1.next()){
240                 //gs.log('JLM - Querying SD Job - ' + gr1.number + "| sysID = " + gr1.sys_id);
241                 var gr2 = new GlideRecord('metric_instance');
242                 gr2.addQuery('id', gr1.sys_id);
243                 gr2.addQuery('u_imported_field', 'Page Count');
244                 gr2.query();
245                 //gs.log(gr2.getRowCount() + " - JLM - This is the Row Count");
246                 if (gr2.getRowCount() < 2){
247                     while(gr2.next()){
248                         if (gr2.u_exception == ''){
249                             //gs.log('JLM - Creating New Page Count Metric for SD Job - ' + gr1.number + "| sysID = " + gr1.sys_id);
250                             gr2.initialize();
251                             gr2.table = 'u_service_delivery_job';
252                             gr2.id = gr1.sys_id.toString();
253                             gr2.u_imported_field = "Page Count";
254                             gr2.u_billable_item_code = "PROD.IMAGE";
255                             gr2.u_exception = 'true';
256                             gr2.insert();
257                         }
258                         else {
259                             //gs.log('JLM - SD Job - ' + gr1.number + ' already has metric created');
260                         }
261                     }
262                 }
263                 else {
264                     //gs.log('JLM - SD Job - ' + gr1.number + ' already has metric created');
265                 }
266             }
267         }
268     },
269
270     populateValue: function () {
271         var gr = new GlideRecord('metric_instance');
272         gr.addQuery('u_imported_field', 'Page Count');
273         gr.addQuery('u_exception', 'true');
274         gr.query();
275     },
276

```

Figure 82. AIM_getMetricBilling.

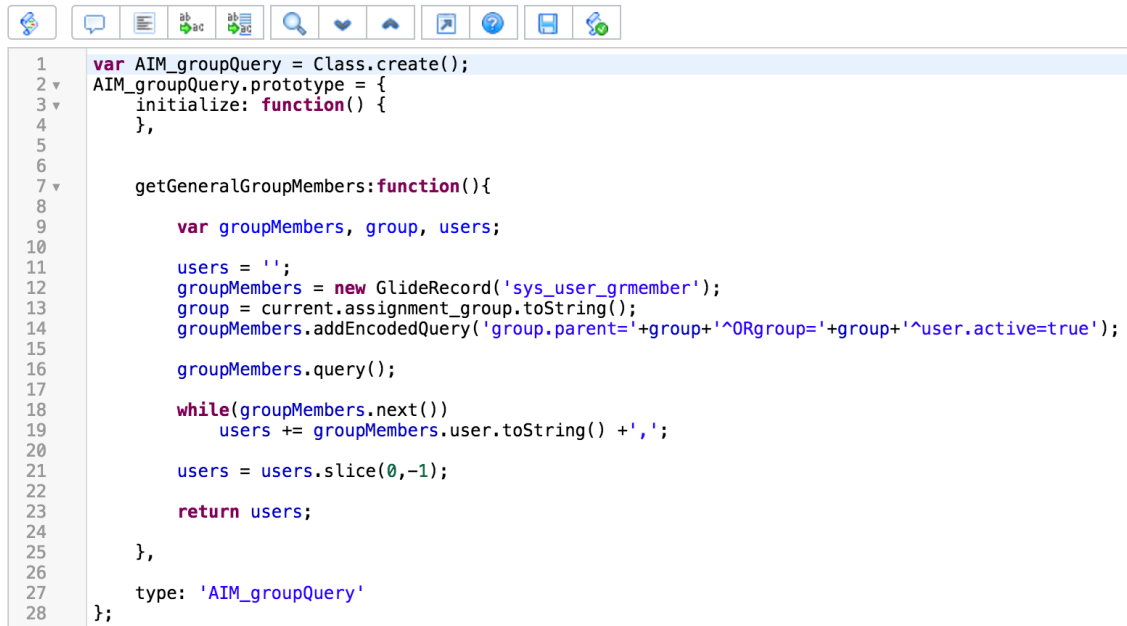
```

272     populateValue: function () {
273         var gr = new GlideRecord('metric_instance');
274         gr.addQuery('u_imported_field', 'Page Count');
275         gr.addQuery('u_exception', 'true');
276         gr.query();
277         while(gr.next()){
278             //gs.log('JLM - Current Metric ID - ' + gr.id);
279             gr.u_value = this.getMetricValue(gr.id);
280             gr.update();
281         }
282     },
283
284     getMetricValue: function (id){
285         var gr = new GlideRecord('metric_instance');
286         gr.addQuery('u_imported_field', 'Page Count');
287         gr.addQuery('u_exception', 'false');
288         gr.addQuery('id', id);
289         gr.query();
290         while(gr.next()){
291             //gs.log('JLM - Current Metric ID - ' + gr.id + " with value - " + gr.u_value);
292             return gr.u_value;
293         }
294     },
295
296 };
297

```

Figure 83. AIM_getMetricBilling.

2) AIM_groupQuery



```

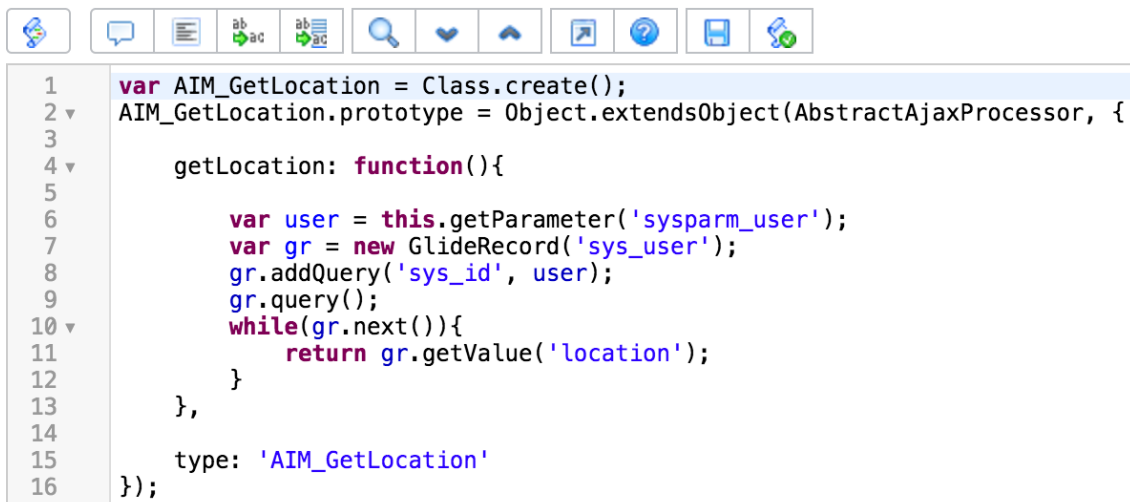
1  var AIM_groupQuery = Class.create();
2  AIM_groupQuery.prototype = {
3  initialize: function() {
4  },
5
6
7  getGeneralGroupMembers: function(){
8
9      var groupMembers, group, users;
10
11     users = '';
12     groupMembers = new GlideRecord('sys_user_grmember');
13     group = current.assignment_group.toString();
14     groupMembers.addEncodedQuery('group.parent='+group+'^ORgroup='+group+'^user.active=true');
15
16     groupMembers.query();
17
18     while(groupMembers.next())
19         users += groupMembers.user.toString() + ',';
20
21     users = users.slice(0,-1);
22
23     return users;
24
25 },
26
27 type: 'AIM_groupQuery'
28 };

```

Figure 84. AIM_groupQuery.

3) AIM_GetLocation

Gets the location of the user record passed as a parameter.



```

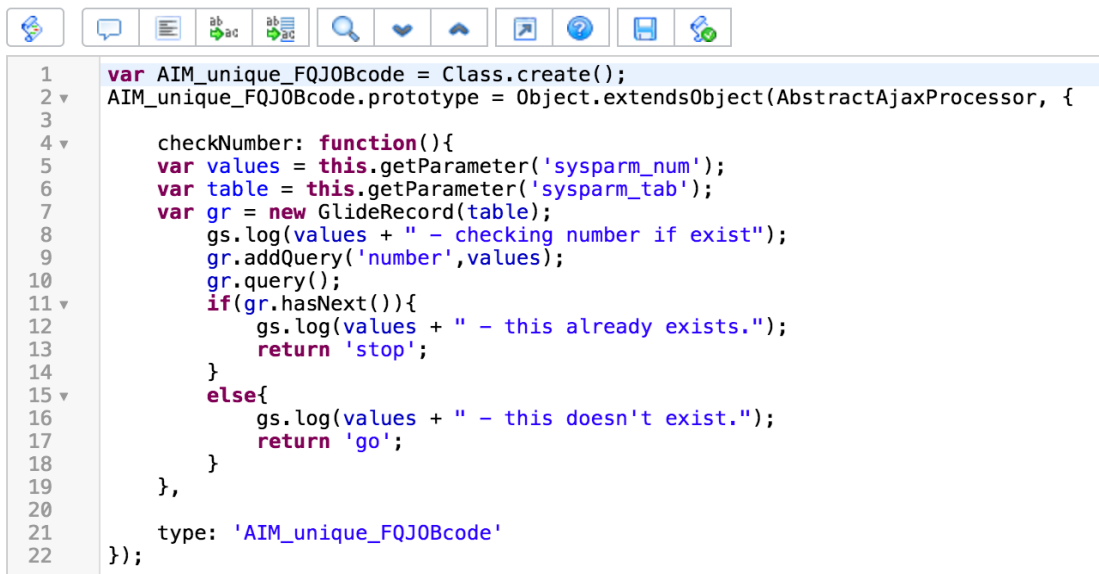
1  var AIM_GetLocation = Class.create();
2  AIM_GetLocation.prototype = Object.extend(Object, AbstractAjaxProcessor, {
3
4  getLocation: function(){
5
6      var user = this.getParameter('sysparm_user');
7      var gr = new GlideRecord('sys_user');
8      gr.addQuery('sys_id', user);
9      gr.query();
10     while(gr.next()){
11         return gr.getValue('location');
12     }
13 },
14
15 type: 'AIM_GetLocation'
16 });

```

Figure 85. AIM_GetLocation.

4) AIM_unique_FQJOBcode

This script makes sure that the job code inserted is unique and there exists no other job with the same FQ Job Code.



```

1  var AIM_unique_FQJOBcode = Class.create();
2  AIM_unique_FQJOBcode.prototype = Object.extend(Object.prototype, {
3
4  checkNumber: function(){
5    var values = this.getParameter('sysparm_num');
6    var table = this.getParameter('sysparm_tab');
7    var gr = new GlideRecord(table);
8    gs.log(values + " - checking number if exist");
9    gr.addQuery('number', values);
10   gr.query();
11   if(gr.hasNext()){
12     gs.log(values + " - this already exists.");
13     return 'stop';
14   }
15   else{
16     gs.log(values + " - this doesn't exist.");
17     return 'go';
18   }
19 },
20
21 type: 'AIM_unique_FQJOBcode'
22 });

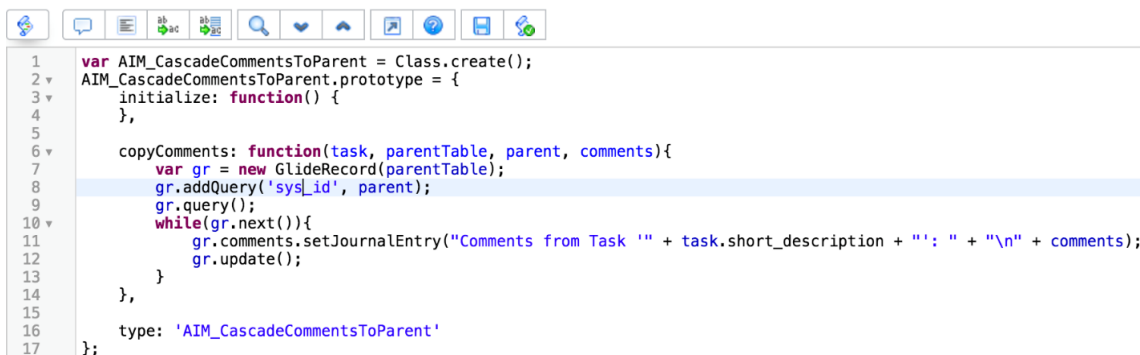
```

Figure 86. AIM_unique_FQJOBcode.

5) AIM_CascadeCommentsToParent

This is to copy comments from the tasks to its parent requests.

Created for copying Intake Task comments to Parent Intake Request, but to be re-usable for other tasks as well.



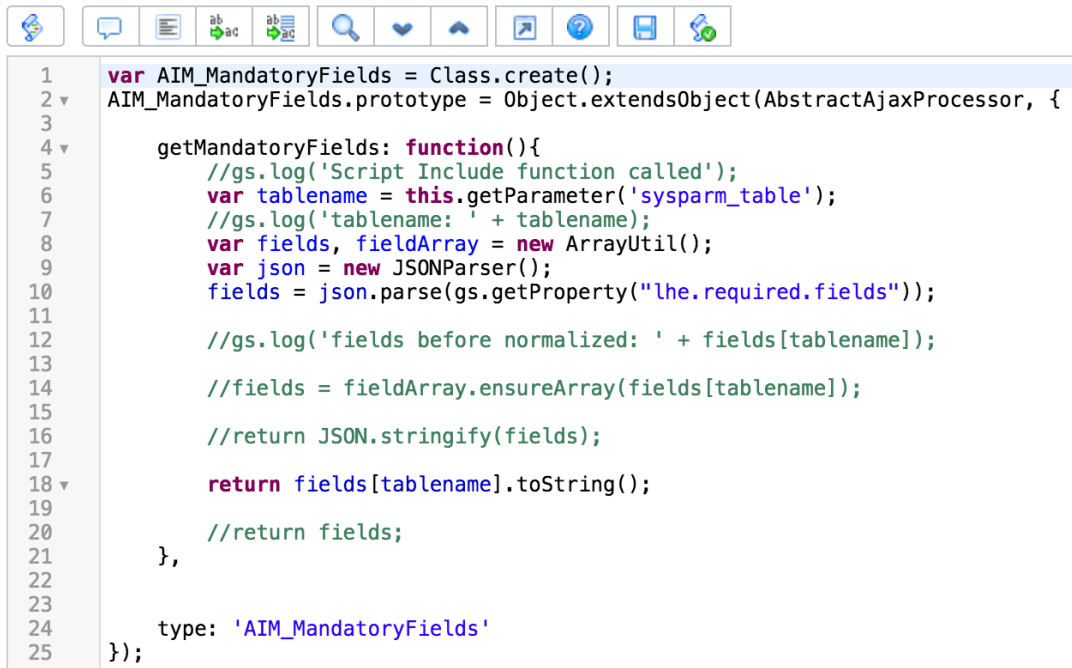
```

1  var AIM_CascadeCommentsToParent = Class.create();
2  AIM_CascadeCommentsToParent.prototype = {
3  initialize: function() {
4  },
5
6  copyComments: function(task, parentTable, parent, comments){
7    var gr = new GlideRecord(parentTable);
8    gr.addQuery('sys_id', parent);
9    gr.query();
10   while(gr.next()){
11     gr.comments.setJournalEntry("Comments from Task '" + task.short_description + "': " + "\n" + comments);
12     gr.update();
13   }
14 },
15
16 type: 'AIM_CascadeCommentsToParent'
17 };

```

Figure 87. AIM_CascadeCommentsToParent.

6) AIM_MandatoryFields



```

1  var AIM_MandatoryFields = Class.create();
2  AIM_MandatoryFields.prototype = Object.extend(Object.prototype, {
3
4  getMandatoryFields: function(){
5      //gs.log('Script Include function called');
6      var tablename = this.getParameter('sysparm_table');
7      //gs.log('tablename: ' + tablename);
8      var fields, fieldArray = new ArrayUtil();
9      var json = new JSONParser();
10     fields = json.parse(gs.getProperty("lhe.required.fields"));
11
12     //gs.log('fields before normalized: ' + fields[tablename]);
13
14     //fields = fieldArray.ensureArray(fields[tablename]);
15
16     //return JSON.stringify(fields);
17
18     return fields[tablename].toString();
19
20     //return fields;
21 },
22
23
24     type: 'AIM_MandatoryFields'
25 });

```

Figure 88. AIM_MandatoryFields.

This script gets all the fields listed in the property 'lhe.required.fields' in JSON form, and returns them to the UI Action 'Spec Ready' discussed in the UI Action section of this paper.

7) lheConditionHolder

This script include is utilized by UI actions' condition field and holds all conditions needed to prevent from extending a table to various sizes and allows for these conditions to be passed or extended.


```

Script
1  var lhcConditionHolder = Class.create();
2  lhcConditionHolder.prototype = {
3  initialize: function() {
4      this.tableCheck = true;
5      this.condition = true;
6  },
7
8  checkShortDesc: function(){
9      var match = false, shortDescArray = ['Final Check', 'QC'], sd = current.short_description;
10     //gs.addInfoMessage(sd);
11     for(i=0; i<shortDescArray.length; i++){
12         if(sd.indexOf(shortDescArray[i]) > -1){
13             //gs.addInfoMessage(shortDescArray[i]);
14             match = true;
15         }
16     }
17     return match;
18 },
19
20 checkShortDescForComplete: function(){
21     var match = false, shortDescArr = ['Provide Specifications'], sd = current.short_description;
22     //gs.addInfoMessage(sd);
23     for(i=0; i<shortDescArr.length; i++){
24         if(sd == (shortDescArr[i])){
25             //gs.addInfoMessage(shortDescArray[i]);
26             match = true;
27         }
28     }
29     return match;
30 },

```

Figure 89. lhcConditionHolder.

```

32 //This is to display Cancel button on active SD Jobs, created in InSite in certain states
33 cancelSDButton: function(){
34     this.condition = (current.active == true) && (current.correlation_id == '');
35     //0, -4 for intake req
36     if(current.sys_class_name == 'u_intake_request'){
37         this.condition = this.condition && (current.state == 0 || current.state == -4 || current.state == 1);
38     }
39     else{
40         this.condition = this.condition && current.state == 1;
41     }
42     return this.condition;
43 },
44
45
46 //This is to display Spec Ready button on Jobs using Job Initiation workflow
47 sdSpecReady: function(){
48     /*this.tableCheck = this.tableExtensionCheck("u_service_delivery_job");
49     this.condition = this.tableCheck && current.state == 1 && current.sys_class_name != 'u_intake_request';
50     return this.condition;*/
51
52     var tables, tableArray;
53     tableArray = new ArrayUtil();
54     tables = JSON.parse(gs.getProperty("lhc.job.include"));
55     tables = tableArray.ensureArray(tables);
56
57     if(parseInt(tableArray.indexOf(tables, current.sys_class_name.toString())) >= 0){
58         this.condition = true;
59     }
60     else this.condition = false;
61     this.condition = this.condition || current.sys_class_name == 'u_production_request';
62     this.condition = this.condition || current.sys_class_name == 'u_imaging_request';
63     //gs.log('Cond before state check: ' + this.condition);
64     this.condition = this.condition && (current.state == 1);
65     //gs.log('Cond after state check: ' + this.condition);
66     return this.condition;
67 },
68
69 //This is to display the Create Quality Incident Form link
70 createQINC: function(){
71     this.tableCheck = this.tableExtensionCheck("u_service_delivery_task");
72     this.tableCheck = this.tableCheck || this.tableExtensionCheck("u_intake_task");
73     this.tableCheck = this.tableCheck || this.tableExtensionCheck("u_matter_task");
74     this.tableCheck = this.tableCheck || this.tableExtensionCheck("u_service_delivery_job");
75     return this.tableCheck;
76 },
77
78 serviceDeliveryQC: function(){
79     //(current.short_description.indexOf('Final Check') > -1 || current.short_description.indexOf('QC') > -1) && current.sys_class
current.state != 4 && current.state != -5
80
81     //(current.short_description.indexOf('Final Check') > -1 || current.short_description.indexOf('QC') > -1) && current.sys_class
current.state != 4 && current.state != -5
82     this.tableCheck = this.tableExtensionCheck("u_service_delivery_task");
83     this.tableCheck = this.tableCheck || this.tableExtensionCheck("u_intake_task");
84     this.condition = this.checkShortDesc() && this.tableCheck && current.state != 3 && current.state != 4 && current.state != -5;
85     return this.condition;
86 },

```

Figure 90. lhcConditionHolder.

```

88 * serviceDeliveryInput:function(){
89 //current.state == 2 || current.state == 1) && current.sys_class_name == 'u_intake_task' && current.short_description.indexOf('Final Chec
current.short_description.indexOf('QC') == -1
90
91 var tableCheck, condition;
92 tableCheck = this.tableExtensionCheck("u_service_delivery_task");
93 tableCheck = tableCheck || this.tableExtensionCheck("u_intake_task");
94 //gs.addInfoMessage('tableCheck: ' + tableCheck + ' SD Check: ' + !this.checkShortDesc() + ' statecheck: ' + (current.state == 2 || curren
95
96 condition = (current.state == 2 || current.state == 1) && tableCheck && !this.checkShortDesc();
97 return condition;
98
99 },
100 * tableExtensionCheck:function(table){
101 var condition;
102 table = new TableUtils(table).getHierarchy();
103 condition = table.indexOf(current.getTable_name().toString()) >= 0;
104 return condition;
105
106 },
107 * serviceDeliveryInputProvided:function(){
108
109 //current.state == 0 && current.sys_class_name == 'u_intake_task'
110 var table;
111
112 table = this.tableExtensionCheck("u_intake_task");
113 table = table || this.tableExtensionCheck("u_service_delivery_task");
114 return table && current.state == 0;
115
116 * serviceDeliveryComplete:function(){
117 var conditionAnswer, tableLadder;
118 //current.state != 0 && current.state != 3 && current.sys_class_name == 'u_intake_task' && current.short_description != 'Intake Request Fi
'QC ESI Intake'
119
120 tableLadder = this.tableExtensionCheck("u_service_delivery_task");
121 tableLadder = tableLadder || this.tableExtensionCheck("u_intake_task");
122
123 conditionAnswer = current.state != 0 && current.state != 3 && tableLadder && !this.checkShortDescForComplete() && !this.checkShortDesc();
124
125 return conditionAnswer;
126
127 },
128 * serviceDeliveryInProgress:function(){
129
130 this.tableCheck = this.tableExtensionCheck("u_intake_task");
131 this.tableCheck = this.tableCheck || this.tableExtensionCheck("u_service_delivery_task");
132
133 //this.condition = current.state == 1 && !this.checkShortDesc() && this.tableCheck;
134 this.condition = current.state == 1 && this.tableCheck;
135
136 return this.condition;
137
138 },
139

```

Figure 91. IheConditionHolder.

```

140 * serviceDeliveryUnassignable:function(){
141 //function will return true when active == true and there is a user assigned
142
143 this.tableCheck = this.tableExtensionCheck("u_intake_task");
144 this.tableCheck = this.tableCheck || this.tableExtensionCheck("u_service_delivery_task");
145
146 //this.condition = current.active == 1 && !this.checkShortDesc() && this.tableCheck && current.assigned_to != '';
147 this.condition = current.active == 1 && this.tableCheck && current.assigned_to != '';
148
149 return this.condition;
150
151 },
152
153 * currentJobList:function(table){
154
155 var productionRequests, productionArray;
156
157 productionArray = [];
158
159 productionRequests = new GlideAggregate(table);
160
161 productionRequests.addEncodedQuery("parent="+current.parent);
162
163 productionRequests.addAggregate("count", "sys_id");
164
165 productionRequests.query();
166
167 * while(productionRequests.next()){
168
169 productionArray.push(productionRequests.sys_id.toString());
170
171 }
172
173 //sys_idINa
174 return "sys_idIN"+productionArray.toString();
175
176 },
177 * //only does reference searches of the parent record.
currentRecordList:function(table, field, key){
178 var recordList, recordArray;
179 gs.log("running current record");
180
181 recordList = new GlideAggregate(table);
182 recordArray = [];
183 * recordList.addEncodedQuery(field+"="+current[key]);
184 recordList.addAggregate("COUNT", "sys_id");
185 recordList.query();
186
187
188 while(recordList.next())
189 recordArray.push(recordList.sys_id.toString());
190
191 return "sys_idIN"+recordArray.toString();
192
193 },
194 type: 'IheConditionHolder'
195 };

```

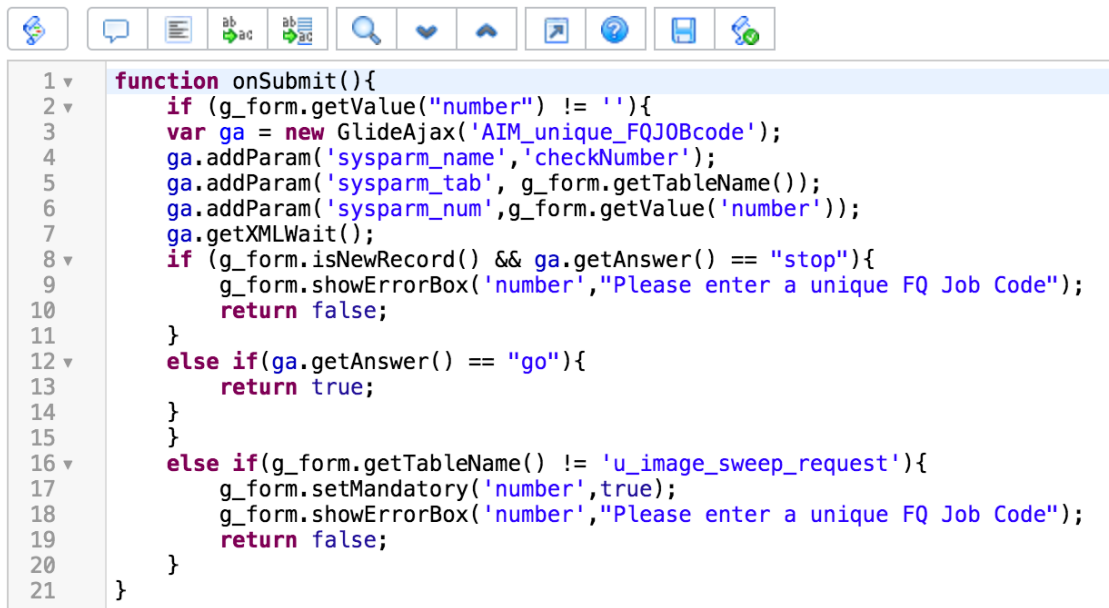
Figure 92. IheConditionHolder.

Client scripts. We'll start with the Client Script created for Service Delivery

Job table.

1) Check Unique FQJOB Code (onSubmit)

This client script calls the Script Include that checks for the uniqueness of the FQ Job code before the record is submitted.



```

1  function onSubmit(){
2  if (g_form.getValue("number") != ''){
3  var ga = new GlideAjax('AIM_unique_FQJOBcode');
4  ga.addParam('sysparm_name','checkNumber');
5  ga.addParam('sysparm_tab', g_form.getTableName());
6  ga.addParam('sysparm_num',g_form.getValue('number'));
7  ga.getXMLWait();
8  if (g_form.isNewRecord() && ga.getAnswer() == "stop"){
9  g_form.showErrorBox('number',"Please enter a unique FQ Job Code");
10 return false;
11 }
12 else if(ga.getAnswer() == "go"){
13 return true;
14 }
15 }
16 else if(g_form.getTableName() != 'u_image_sweep_request'){
17 g_form.setMandatory('number',true);
18 g_form.showErrorBox('number',"Please enter a unique FQ Job Code");
19 return false;
20 }
21 }

```

Figure 93. Check unique FQ job code.

2) Hide annotations if Asset (onChange)

```

Script
1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2      if (isLoading) {
3          return;
4      }
5      if(g_form.getTableId() == 'u_imaging_request'){
6          if(newValue == 'Relativity'){
7              document.getElementById("anno1").parentNode.style.display="block";
8              document.getElementById("anno4").parentNode.style.display="block";
9              document.getElementById("anno5").parentNode.style.display="block";
10             document.getElementById("anno7").parentNode.style.display="block";
11         }
12     }
13     else if(newValue == ""){
14         document.getElementById("anno1").parentNode.style.display="none";
15         document.getElementById("anno4").parentNode.style.display="none";
16         document.getElementById("anno5").parentNode.style.display="none";
17         document.getElementById("anno7").parentNode.style.display="none";
18     }
19 }
20 else{
21     if(getView() == 'hs_view'){
22         if(newValue == "Relativity"){
23             document.getElementById("anno1").parentNode.style.display="block";
24             document.getElementById("anno3").parentNode.style.display="block";
25             document.getElementById("anno4").parentNode.style.display="block";
26             document.getElementById("anno5").parentNode.style.display="block";
27             document.getElementById("anno8").parentNode.style.display="block";
28             document.getElementById("anno9").parentNode.style.display="block";
29             document.getElementById("anno10").parentNode.style.display="block";
30             document.getElementById("anno11").parentNode.style.display="block";
31             document.getElementById("anno12").parentNode.style.display="block";
32             document.getElementById("anno13").parentNode.style.display="block";
33             document.getElementById("anno14").parentNode.style.display="block";
34         }
35     }
36     else{
37         document.getElementById("anno1").parentNode.style.display="none";
38         document.getElementById("anno3").parentNode.style.display="none";
39         document.getElementById("anno4").parentNode.style.display="none";
40         document.getElementById("anno5").parentNode.style.display="none";
41         document.getElementById("anno8").parentNode.style.display="none";
42         document.getElementById("anno9").parentNode.style.display="none";
43         document.getElementById("anno10").parentNode.style.display="none";
44         document.getElementById("anno11").parentNode.style.display="none";
45         document.getElementById("anno12").parentNode.style.display="none";
46         document.getElementById("anno13").parentNode.style.display="none";
47         document.getElementById("anno14").parentNode.style.display="none";
48     }
49 }
50 else if(getView() == 'specifications'){
51     if (newValue == "Asset"){
52         document.getElementById("anno1").parentNode.style.display="none";
53         document.getElementById("anno2").parentNode.style.display="none";
54         document.getElementById("anno3").parentNode.style.display="none";
55         document.getElementById("anno4").parentNode.style.display="none";
56         document.getElementById("anno5").parentNode.style.display="none";
57         document.getElementById("anno6").parentNode.style.display="block";
58         document.getElementById("anno7").parentNode.style.display="none";
59         document.getElementById("anno8").parentNode.style.display="none";
60     }
61     else if(newValue == "Relativity"){
62         document.getElementById("anno1").parentNode.style.display="block";
63         document.getElementById("anno2").parentNode.style.display="block";
64         document.getElementById("anno3").parentNode.style.display="block";
65         document.getElementById("anno4").parentNode.style.display="block";
66         document.getElementById("anno5").parentNode.style.display="block";
67         document.getElementById("anno6").parentNode.style.display="none";
68         document.getElementById("anno7").parentNode.style.display="block";
69         document.getElementById("anno8").parentNode.style.display="block";
70     }
71 }

```

Figure 94. Hide annotations if asset.

```

70 ▾      else if(newValue == ""){
71          document.getElementById("anno1").parentNode.style.display="none";
72          document.getElementById("anno2").parentNode.style.display="none";
73          document.getElementById("anno3").parentNode.style.display="none";
74          document.getElementById("anno4").parentNode.style.display="none";
75          document.getElementById("anno5").parentNode.style.display="none";
76          document.getElementById("anno6").parentNode.style.display="none";
77          document.getElementById("anno7").parentNode.style.display="none";
78          document.getElementById("anno8").parentNode.style.display="none";
79      }
80  }
81 }
82 }

```

Figure 95. Hide annotations if asset.

This script shows/hides the Annotations on the Production Request table, based on the Data source field value.

3) Autofill Suffix on PR (OnChange)



```

1 ▾ function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2 ▾   if (isLoading || newValue === '') {
3     return;
4   }
5
6   var regex = /^[^\\\/\:\#\*\?'\u2012\u2019\u02BC\u201C\u201D\u201E\u201F\u2012\u2013\u2014\u2015\u007c<>\\|]*$/;
7   var ans = g_form.getValue('u_static_endorsement');
8 ▾   if(!regex.test(ans)){
9     //alert('Please enter valid string for "Static Endorsement"');
10    //return;
11  }
12 ▾   else{
13     var suffix = newValue.replace(/\n/g, ' ');
14     g_form.setValue('u_native_renaming_suffix', suffix);
15     //g_form.setValue('u_native_renaming_suffix', newValue);
16   }
17 }
18 }

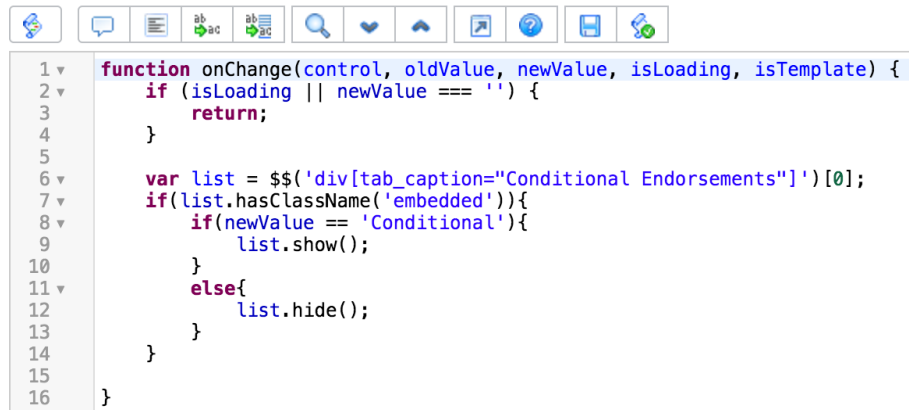
```

Figure 96. Autofill suffix on PR.

This script makes sure that the field 'Static Endorsement' contains a value without invalid characters.

4) Show/Hide Endorsements (onChange)

This script displays the 'Conditional Endorsements' embedded list when the 'Confidentiality Mapping' field contains Conditional.



```

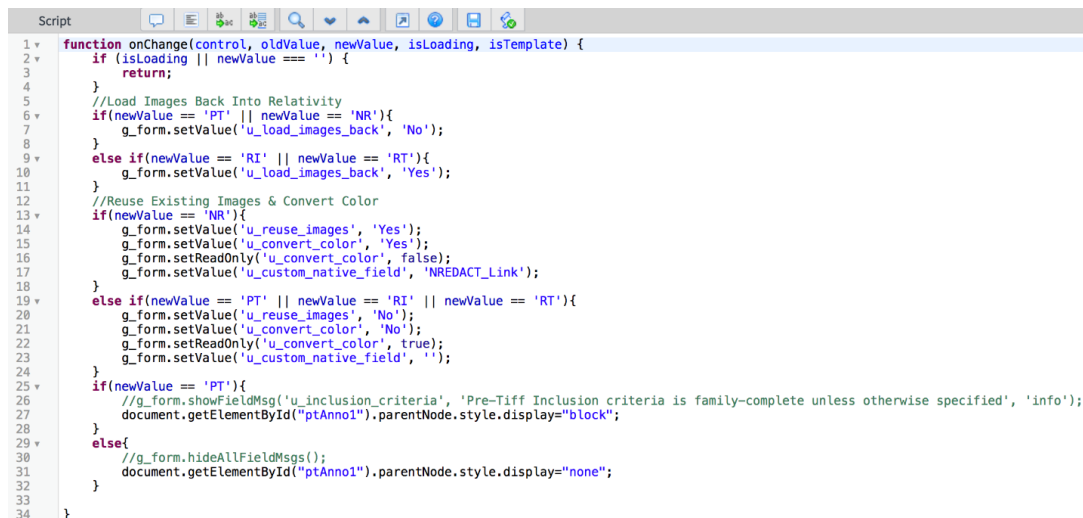
1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2      if (isLoading || newValue === '') {
3          return;
4      }
5
6      var list = $$('div[tab_caption="Conditional Endorsements"]')[0];
7      if(list.hasClass('embedded')){
8          if(newValue == 'Conditional'){
9              list.show();
10         }
11         else{
12             list.hide();
13         }
14     }
15 }
16 }

```

Figure 97. Show/hide endorsements.

5) Choices based on Sweep Type (OnChange)

This script runs on the Image Sweep Request table.



```

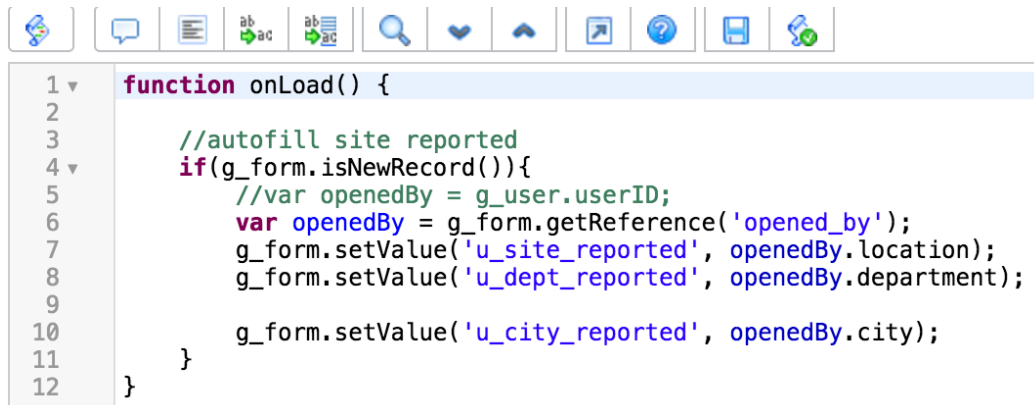
Script
1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2      if (isLoading || newValue === '') {
3          return;
4      }
5      //Load Images Back Into Relativity
6      if(newValue == 'PT' || newValue == 'NR'){
7          g_form.setValue('u_load_images_back', 'No');
8      }
9      else if(newValue == 'RI' || newValue == 'RT'){
10         g_form.setValue('u_load_images_back', 'Yes');
11     }
12     //Reuse Existing Images & Convert Color
13     if(newValue == 'NR'){
14         g_form.setValue('u_reuse_images', 'Yes');
15         g_form.setValue('u_convert_color', 'Yes');
16         g_form.setReadOnly('u_convert_color', false);
17         g_form.setValue('u_custom_native_field', 'NREDACT_Link');
18     }
19     else if(newValue == 'PT' || newValue == 'RI' || newValue == 'RT'){
20         g_form.setValue('u_reuse_images', 'No');
21         g_form.setValue('u_convert_color', 'No');
22         g_form.setReadOnly('u_convert_color', true);
23         g_form.setValue('u_custom_native_field', '');
24     }
25     if(newValue == 'PT'){
26         //g_form.showFieldMsg('u_inclusion_criteria', 'Pre-Tiff Inclusion criteria is family-complete unless otherwise specified', 'info');
27         document.getElementById("ptAnnoi").parentNode.style.display="block";
28     }
29     else{
30         //g_form.hideAllFieldMsgs();
31         document.getElementById("ptAnnoi").parentNode.style.display="none";
32     }
33 }
34 }

```

Figure 98. Choices based on sweep type.

This script changes the values of various fields like 'Load Images back into Relativity', 'Convert color to black & white', 'Custom Native field' etc. based on the Sweep Type chosen.

6) Autofill Dept. & Site Reported (onLoad)



```

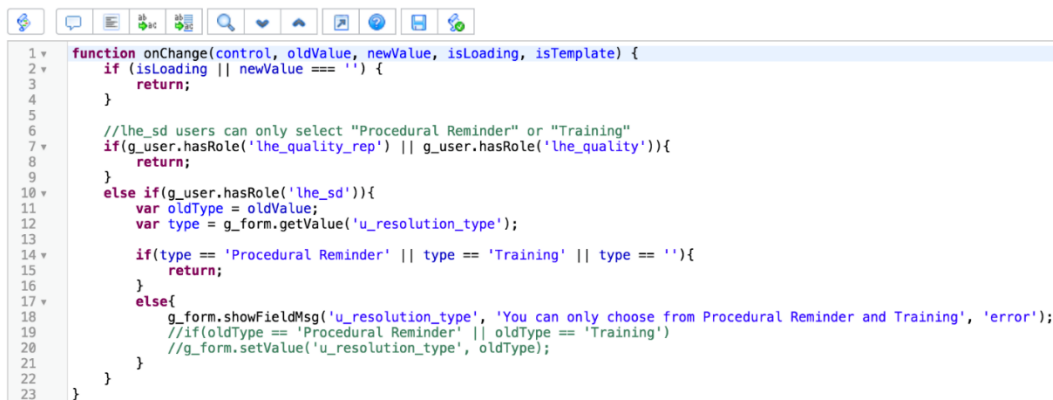
1  function onLoad() {
2
3      //autofill site reported
4  if(g_form.isNewRecord()){
5      //var openedBy = g_user.userID;
6      var openedBy = g_form.getReference('opened_by');
7      g_form.setValue('u_site_reported', openedBy.location);
8      g_form.setValue('u_dept_reported', openedBy.department);
9
10     g_form.setValue('u_city_reported', openedBy.city);
11 }
12 }

```

Figure 99. Autofill dept. & site reported.

This script sets the 'Site Reported', 'Dept. reported' and 'City reported' fields to the respective values from the 'Opened by' field on a new record.

7) Resolution Type options for lhe_sd (onChange)



```

1  function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2  if (isLoading || newValue === '') {
3      return;
4  }
5
6  //lhe_sd users can only select "Procedural Reminder" or "Training"
7  if(g_user.hasRole('lhe_quality_rep') || g_user.hasRole('lhe_quality')){
8      return;
9  }
10 else if(g_user.hasRole('lhe_sd')){
11     var oldType = oldValue;
12     var type = g_form.getValue('u_resolution_type');
13
14     if(type == 'Procedural Reminder' || type == 'Training' || type == ''){
15         return;
16     }
17     else{
18         g_form.showFieldMsg('u_resolution_type', 'You can only choose from Procedural Reminder and Training', 'error');
19         //if(oldType == 'Procedural Reminder' || oldType == 'Training')
20         //g_form.setValue('u_resolution_type', oldType);
21     }
22 }
23 }

```

Figure 100. Resolution type options.

This script restricts the users without lhe_quality_rep or lhe_quality role to choose only from 'Procedural Reminder' or 'Training' for the 'Resolution Type' field.

8) COC & Calculate size in GB (onSubmit)

This script runs on 'Evidence' table.

```

Script
1  function onSubmit() {
2      //Calculate Received size and Extracted size in GB from bytes
3
4      var rgb = 0, egb = 0;
5      var rsize = g_form.getValue('u_received_size');
6      var esize = g_form.getValue('u_extracted_size');
7
8      rsize = rsize.replace(/,/g, '');
9      esize = esize.replace(/,/g, '');
10
11     if(rsize != '')
12         rgb = rsize/1073741824;
13         //rgb = rsize/Math.pow(1024, 3).toFixed(3);
14         //rgb = Math.abs(rgb).toFixed(3);
15
16     else rgb = '';
17     if(esize != '')
18         egb = esize/1073741824;
19     else egb = '';
20
21     g_form.setValue('u_received_size_gb', rgb);
22     g_form.setValue('u_extracted_size_gb', egb);
23
24     //Chain of Custody must be checked if "Media Type" is anything but "Email", "FTP Site", or "Other" is chosen. Error message should be generated on submission if it is not checked.
25
26     if(g_form.getValue('media_type') != 'Email' && g_form.getValue('media_type') != 'FTP Site' && g_form.getValue('media_type') != 'Other'){
27         if(g_form.getValue('coc_generated') == 'false' && !g_user.hasRole('the_sd_pr')){
28             alert('Chain of Custody form generated is not checked.');
```

Figure 101. COC & calculate size in GB.

It calculates the Received and extracted sizes in GB based on their respective values in bytes.

Also, makes sure Chain of Custody is checked if "Media Type" is anything but "Email", "FTP Site", or "Other" is chosen. Error message is generated on submission if it is not checked.

9) Check Attachments (onSubmit)

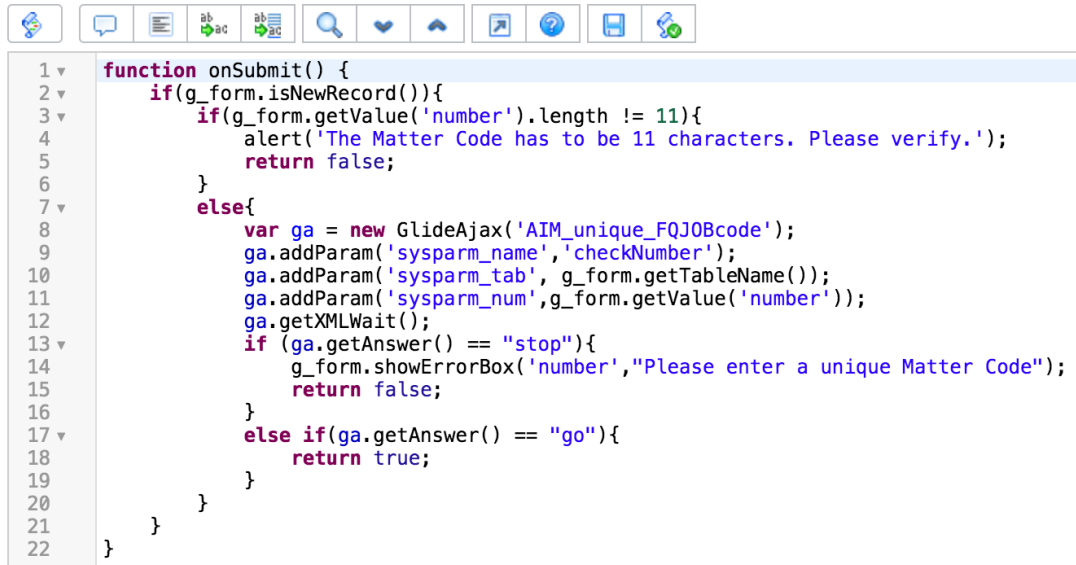
This runs on Media Notification table. If 'Documentation included' is checked, this script verifies if there's at least one attachment to the form before submitting the form.

```

1  function onSubmit() {
2      //If Documentation included is checked, this script verifies if there's atleast one attachment to the form before
3      submitting
4      var sysID = g_form.getUniqueValue();
5      var doc = g_form.getValue('u_doc_included');
6      if(doc == "true"){
7          var gr = new GlideRecord("sys_attachment");
8          gr.addQuery("table_name", "u_media_notification");
9          gr.addQuery("table_sys_id", sysID);
10         gr.query();
11         if (!gr.next()) {
12             alert("You must attach a file to submit.");
13             return false;
14         }
15     }
16 }
```

Figure 102. Check attachments.

10) Validate Matter code (onSubmit)



```

1  function onSubmit() {
2      if(g_form.isNewRecord()){
3          if(g_form.getValue('number').length != 11){
4              alert('The Matter Code has to be 11 characters. Please verify.');
```

Figure 103. Validate matter code.

This runs on the Matter table and verifies that the Matter code is according to the criteria. It also makes sure that the Matter code is unique by utilizing the Script Include.

11) Default choices based on Type field (onChange)

This script runs on the Placeholder Specifications table. It sets the values of various field based on the specification type.

```

Script
1 function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2   if (isLoading || newValue === '') {
3     return;
4   }
5
6   if(newValue == 'excep'){
7     //g_form.setValue('u_name', 'Exception');
8     g_form.setValue('u_slipsheet_lang', 1); //Exception Reason
9     g_form.setValue('u_custom_slipsheet_lang', '');
10    g_form.setValue('u_filename', 'No');
11    g_form.setValue('u_native', 'No');
12    g_form.setValue('u_null_metadata', 'No');
13    g_form.setValue('u_endorse_confidentiality', 'No');
14    g_form.setValue('u_qc_flag', 'Exception');
15  }
16  else if (newValue == 'native'){
17    //g_form.setValue('u_name', 'File Provided Natively');
18    g_form.setValue('u_slipsheet_lang', 2); //Custom Slipsheet Language
19    g_form.setValue('u_custom_slipsheet_lang', 'File Provided Natively');
20    g_form.setValue('u_filename', 'No');
21    g_form.setValue('u_text_file', 2); //Document Text
22    g_form.setValue('u_native', 'Yes');
23    g_form.setValue('u_null_metadata', 'No');
24    g_form.setValue('u_endorse_confidentiality', 'Yes');
25    g_form.setValue('u_qc_flag', 'File Provided Natively');
26  }
27  else if (newValue == 'non_res'){
28    //g_form.setValue('u_name', 'Non-Responsive');
29    g_form.setValue('u_slipsheet_lang', 2); //Custom Slipsheet Language
30    g_form.setValue('u_custom_slipsheet_lang', 'Withheld as Not Responsive');
31    g_form.setValue('u_filename', 'No');
32    g_form.setValue('u_text_file', 1);
33    g_form.setValue('u_native', 'No');
34    g_form.setValue('u_null_metadata', 'Yes');
35    g_form.setValue('u_endorse_confidentiality', 'No');
36    g_form.setValue('u_qc_flag', '[NON-RESP]');
37  }
38  else if (newValue == 'priv'){
39    //g_form.setValue('u_name', 'Privileged');
40    g_form.setValue('u_slipsheet_lang', 2); //Custom Slipsheet Language
41    g_form.setValue('u_custom_slipsheet_lang', 'Withheld for Privilege');
42    g_form.setValue('u_filename', 'No');
43    g_form.setValue('u_native', 'No');
44    g_form.setValue('u_text_file', 1);
45    g_form.setValue('u_null_metadata', 'Yes');
46    g_form.setValue('u_endorse_confidentiality', 'No');
47    g_form.setValue('u_qc_flag', '[PRIV_HIT]');
48  }
49  else if (newValue == 'tech_issue'){
50    //g_form.setValue('u_name', 'Technical Issue');
51    g_form.setValue('u_slipsheet_lang', 2); //Custom Slipsheet Language
52    g_form.setValue('u_custom_slipsheet_lang', 'Technical Issue');
53    g_form.setValue('u_text_file', 1); //Placeholder text
54    g_form.setValue('u_native', 'No');
55    g_form.setValue('u_filename', 'No');
56    g_form.setValue('u_null_metadata', 'Yes');
57    g_form.setValue('u_endorse_confidentiality', 'No');

```

Figure 104. Default choices based on type.

```

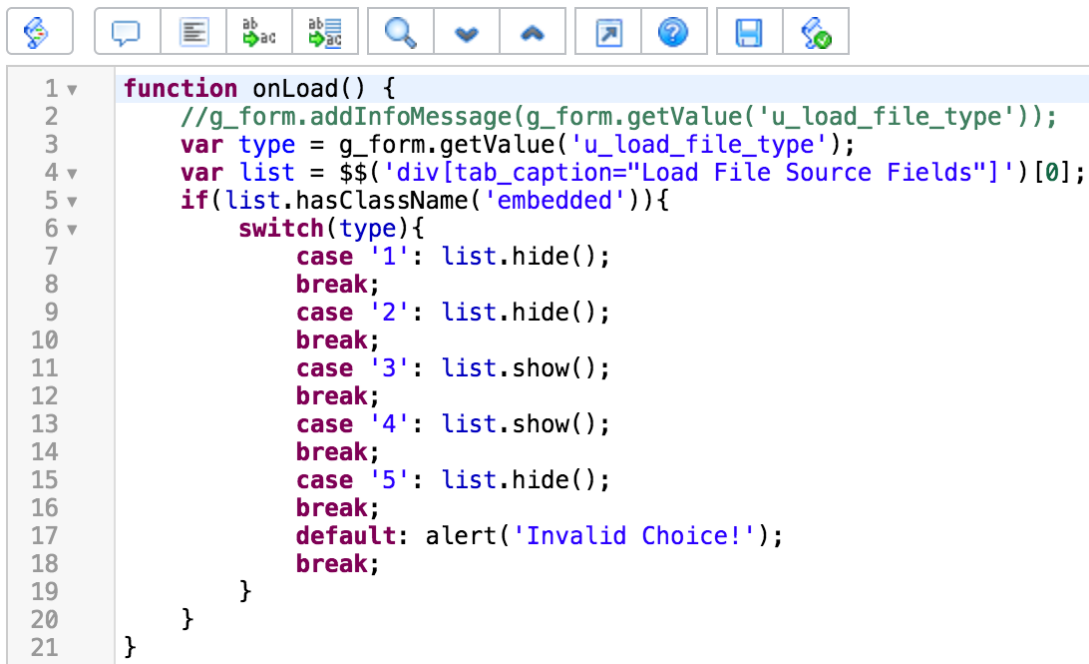
58     g_form.setValue('u_qc_flag', 'Additional Custom Placeholder');
59 }
60 else if (newValue == 'thresh_exc'){
61     g_form.setValue('u_qc_flag', 'Threshold Exceeded');
62 }
63 else if (newValue == 'other'){
64     g_form.setValue('u_custom_slipsheet_lang', '');
65     g_form.setValue('u_filename', 'No');
66     g_form.setValue('u_null_metadata', 'No');
67     g_form.setValue('u_endorse_confidentiality', 'No');
68     g_form.clearValue('u_qc_flag');
69     g_form.showFieldMsg('u_qc_flag', 'QC Flag value to be provided by pod-aligned technician(s)',
70                         'error');
71 }
72 }

```

Figure 105. Default choices based on type.

12) Show/Hide Load File Related list (onLoad)

This script runs on Load File table and shows/hides the Load File Source Fields embedded list based on the Load File type field.



```

1  function onLoad() {
2      //g_form.addInfoMessage(g_form.getValue('u_load_file_type'));
3      var type = g_form.getValue('u_load_file_type');
4      var list = $$('div[tab_caption="Load File Source Fields"]')[0];
5      if(list.hasClassName('embedded')){
6          switch(type){
7              case '1': list.hide();
8                  break;
9              case '2': list.hide();
10                 break;
11             case '3': list.show();
12                 break;
13             case '4': list.show();
14                 break;
15             case '5': list.hide();
16                 break;
17             default: alert('Invalid Choice!');
18                 break;
19         }
20     }
21 }

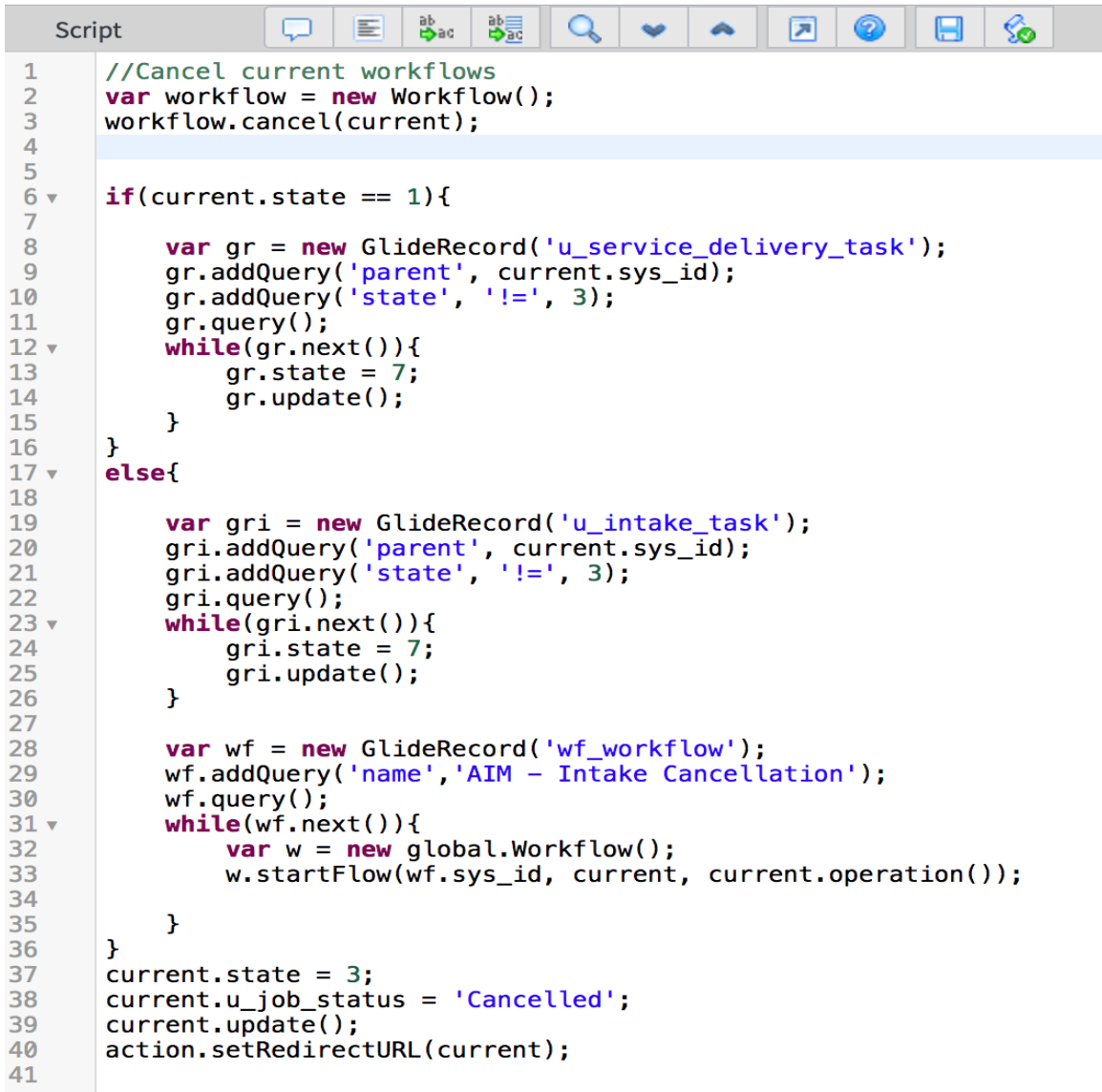
```

Figure 106. Show/hide load file related list.

UI Actions:

The following are the custom UI actions created to implement custom functionality on different tables.

1) Cancel



```

1 //Cancel current workflows
2 var workflow = new Workflow();
3 workflow.cancel(current);
4
5
6 if(current.state == 1){
7
8     var gr = new GlideRecord('u_service_delivery_task');
9     gr.addQuery('parent', current.sys_id);
10    gr.addQuery('state', '!=', 3);
11    gr.query();
12    while(gr.next()){
13        gr.state = 7;
14        gr.update();
15    }
16 }
17 else{
18
19     var gri = new GlideRecord('u_intake_task');
20     gri.addQuery('parent', current.sys_id);
21     gri.addQuery('state', '!=', 3);
22     gri.query();
23     while(gri.next()){
24         gri.state = 7;
25         gri.update();
26     }
27
28     var wf = new GlideRecord('wf_workflow');
29     wf.addQuery('name', 'AIM - Intake Cancellation');
30     wf.query();
31     while(wf.next()){
32         var w = new global.Workflow();
33         w.startFlow(wf.sys_id, current, current.operation());
34     }
35 }
36 }
37 current.state = 3;
38 current.u_job_status = 'Cancelled';
39 current.update();
40 action.setRedirectURL(current);
41

```

Figure 107. Cancel job.

Condition: `new lheConditionHolder().cancelSDButton()`

Condition calls the Script Include to verify the availability of the Cancel button.

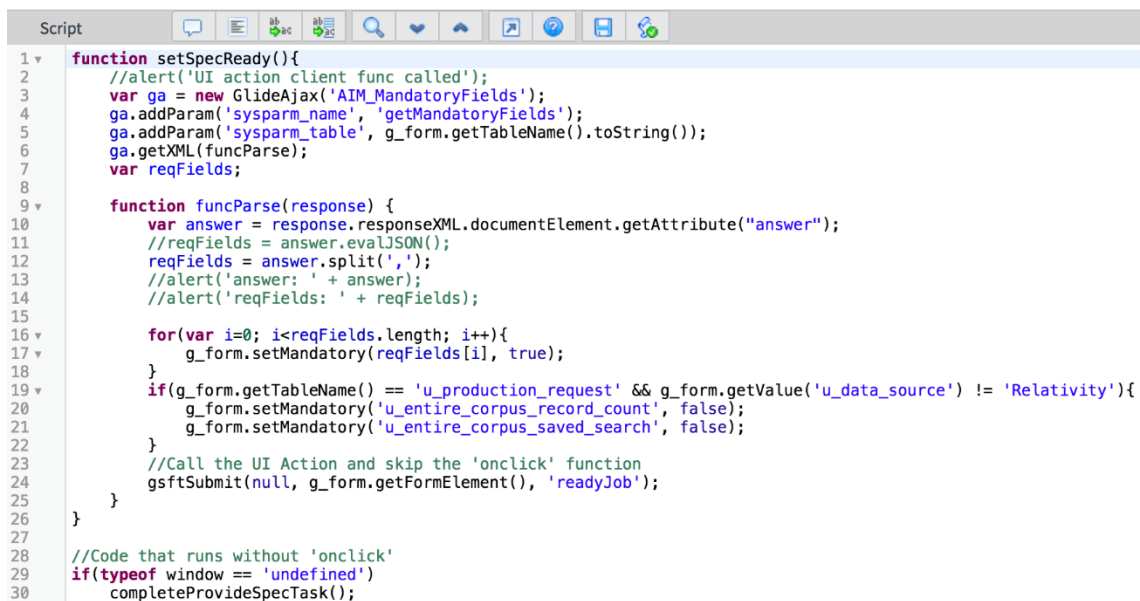
This button is used to cancel a Job, so that users no longer work on it. This is because sometimes requests are created with the wrong data, or the job gets cancelled midway.

This script starts with canceling the workflows currently running on the current job. It then closes all the child tasks that are still open, and sets the Job status to Cancelled. For Intake Requests not in Draft state, it triggers the 'Intake cancellation' workflow.

2) Spec Ready

Condition: `new lhcConditionHolder().sdSpecReady()`

Condition calls the Script Include to verify the availability of the Spec Ready button.



```

1  function setSpecReady(){
2  //alert('UI action client func called');
3  var ga = new GlideAjax('AIM_MandatoryFields');
4  ga.addParam('sysparm_name', 'getMandatoryFields');
5  ga.addParam('sysparm_table', g_form.getTable().toString());
6  ga.getXML(funcParse);
7  var reqFields;
8
9  function funcParse(response) {
10     var answer = response.responseXML.documentElement.getAttribute("answer");
11     //reqFields = answer.evalJSON();
12     reqFields = answer.split(',');
13     //alert('answer: ' + answer);
14     //alert('reqFields: ' + reqFields);
15
16     for(var i=0; i<reqFields.length; i++){
17         g_form.setMandatory(reqFields[i], true);
18     }
19     if(g_form.getTable() == 'u_production_request' && g_form.getValue('u_data_source') != 'Relativity'){
20         g_form.setMandatory('u_entire_corpus_record_count', false);
21         g_form.setMandatory('u_entire_corpus_saved_search', false);
22     }
23     //Call the UI Action and skip the 'onclick' function
24     gsftSubmit(null, g_form.getFormElement(), 'readyJob');
25 }
26 }
27
28 //Code that runs without 'onclick'
29 if(typeof window == 'undefined')
30     completeProvideSpecTask();

```

Figure 108. Spec ready.

This button works on all SD Jobs and calls the Script Include 'Mandatory Fields' to retrieve the mandatory fields on the current table and make them required. Once they are filled, it increments the version by 1 and sets the State to Ready for Production and Imaging Requests. For other jobs, it closes the associated 'Provide Specifications' task and saves the current record.

```

33 ▾ function completeProvideSpecTask(){
34     //current.update();
35     if(current.sys_class_name == 'u_imaging_request' || current.sys_class_name == 'u_production_request'){
36         current.state = 2;
37     }
38     if(current.u_version == ''){
39         current.u_version=1;
40     }
41     current.update();
42     gs.addInfoMessage('Request ' + current.number + ' is now ready.');
```

Figure 109. Spec ready.

3) Complete Check In

This exists on the 'Evidence' table and completes the check-in of the Evidence by making certain fields mandatory. Once done, it checks the 'Check in' field and saves the record.

```

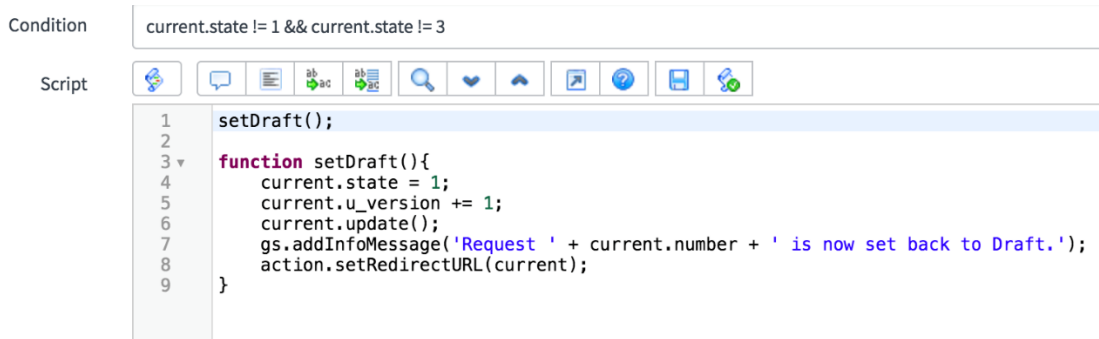
1 //Client-side 'onclick' function
2 ▾ function checkIn(){
3     if(g_form.getValue('media_type') != 'FTP Site' && g_form.getValue('media_type') != 'Email'){
4         g_form.setMandatory('manufacturer', 'true');
5         g_form.setMandatory('model', 'true');
6     }
7     g_form.setMandatory('u_extracted_sizes', 'true');
8     g_form.setMandatory('u_received_size', 'true');
9     g_form.setMandatory('esi_vault', 'true');
10    g_form.setMandatory('esi_location', 'true');
11
12    //Call the UI Action and skip the 'onclick' function
13    gsftSubmit(null, g_form.getFormElement(), 'completeCheckIn');
14 }
15
16 //Code that runs without 'onclick'
17 if(typeof window == 'undefined')
18     serverCheckIn();
19
20 ▾ function serverCheckIn(){
21     //Set the 'Initial Check In Complete?' to 'True', update and reload the record
22     current.initial_check_in = true;
23     current.update();
24     gs.addInfoMessage('Initial Check In Complete for Intake Request: ' + current.number);
25     action.setRedirectURL(current);
26 }
27
```

Figure 110. Complete check in.

4) Back to Draft

This exists on Production Request and its extended tables,

It sets the PR's state back to Draft, increments version number and make most fields editable again.



The screenshot shows a code editor interface. At the top, there is a 'Condition' field containing the text 'current.state != 1 && current.state != 3'. Below this is a 'Script' field with a toolbar containing various icons. The script code is as follows:

```
1 setDraft();
2
3 function setDraft(){
4     current.state = 1;
5     current.u_version += 1;
6     current.update();
7     gs.addInfoMessage('Request ' + current.number + ' is now set back to Draft. ');
8     action.setRedirectURL(current);
9 }
```

Figure 111. Back to draft.

5) Create Metrics

This UI action is available to all SD Jobs to create metrics depending on the Job type, except for Image Sweep request.

```

Script
1 createMetrics();
2
3 function createMetrics(){
4     if(current.u_job_type == 'ESI Review Volume' || current.u_job_type == 'Native Production' ||
5     current.u_job_type == 'Decryption' || current.u_job_type == 'Major Exceptions'){
6         //gs.log("category is " + current.category);
7         var char1 = [];
8         if(current.company == '8c6d9ecb1306d200ff89b168d144b04f'){ //If client is 'Johnson & Johnson'
9             char1 = ["EMAILSMART (Total)", "EMAILSMART SUPPRESSED", "EMAILSMART REMAINDER",
10            "EMAILSMART % REDUCTION", "Document Exceptions - Promoted",
11            "Document Exceptions - Not Promoted", "Major Exceptions",
12            "J&J Nonsubstantive Suppressed", "J&J Remainder for Review"];
13        }
14        else{
15            char1 = ["EMAILSMART (Total)", "EMAILSMART SUPPRESSED", "EMAILSMART REMAINDER",
16            "EMAILSMART % REDUCTION", "Document Exceptions - Promoted",
17            "Document Exceptions - Not Promoted", "Major Exceptions"];
18        }
19
20        for(var i=0; i<char1.length; i++){
21            //gs.log("function called for " + char1[i]);
22            var gri = new GlideRecord('x_led_metrics_job_metric');
23            gri.addQuery('parent', current.getUniqueValue());
24            gri.addQuery('u_data_point', char1[i]);
25            gri.query();
26            if(!gri.hasNext()){
27                createJobMetrics(char1[i]);
28            }
29            else{
30                action.setRedirectURL(current);
31            }
32        }
33    }
34
35    if(current.u_job_type == 'u_production_request' || current.u_job_type == 'Imaging' || current.u_job_type == 'Image w/Native Production'){
36        var char2 = [];
37        if(current.company == '8c6d9ecb1306d200ff89b168d144b04f'){ //If client is 'Johnson & Johnson'
38            char2 = ["Imaging Exceptions", "Produced Natively", "J&J Estimated Native Pages"];
39        }
40        else{
41            char2 = ["Imaging Exceptions", "Produced Natively"];
42        }
43
44        for(var j=0; j<char2.length; j++){
45            var grj = new GlideRecord('x_led_metrics_job_metric');
46            grj.addQuery('parent', current.getUniqueValue());
47            grj.addQuery('u_data_point', char2[j]);
48            grj.query();
49            if(!grj.hasNext()){
50                createJobMetrics(char2[j]);
51            }
52            else{
53                action.setRedirectURL(current);
54            }
55        }
56    }
57 }

```

Figure 112. Create metrics.


```

58 ▾ if(current.u_job_type == 'Reuse' || current.u_job_type == 'Native Redaction'){
59     var char3 = [];
60 ▾   char3 = ["Imaging Exceptions", "Master Doc Count", "Master Page Count",
61           "Duplicate Doc Count", "Duplicate Page Count"];
62 ▾   for(var k=0; k<char3.length; k++){
63       var grk = new GlideRecord('x_led_metrics_job_metric');
64       grk.addQuery('parent', current.getUniqueValue());
65 ▾     grk.addQuery('u_data_point', char3[k]);
66     grk.query();
67 ▾     if(!grk.hasNext()){
68 ▾       createJobMetrics(char3[k]);
69     }
70 ▾     else{
71       action.setRedirectURL(current);
72     }
73   }
74 }
75
76 ▾ if(current.u_job_type == 'Custom Request'){
77     var char4 = [];
78 ▾     if(current.company == '8c6d9ecb1306d200ff89b168d144b04f'){ //If client is 'Johnson & Johnson'
79 ▾       char4 = ["EMAILSMART (Total)", "EMAILSMART SUPPRESSED", "EMAILSMART REMAINDER",
80             "EMAILSMART % REDUCTION", "Document Exceptions - Promoted",
81             "Document Exceptions - Not Promoted", "Major Exceptions", "Imaging Exceptions",
82             "Produced Natively", "Master Doc Count", "Master Page Count",
83             "Duplicate Doc Count", "Duplicate Page Count",
84             "J&J_Nonsubstantive Suppressed", "J&J_Remainder for Review"];
85     }
86 ▾     else{
87 ▾       char4 = ["EMAILSMART (Total)", "EMAILSMART SUPPRESSED", "EMAILSMART REMAINDER",
88             "EMAILSMART % REDUCTION", "Document Exceptions - Promoted",
89             "Document Exceptions - Not Promoted", "Major Exceptions", "Imaging Exceptions",
90             "Produced Natively", "Master Doc Count", "Master Page Count",
91             "Duplicate Doc Count", "Duplicate Page Count"];
92     }
93 ▾     for(var l=0; l<char4.length; l++){
94       var grl = new GlideRecord('x_led_metrics_job_metric');
95       grl.addQuery('parent', current.getUniqueValue());
96 ▾     grl.addQuery('u_data_point', char4[l]);
97     grl.query();
98 ▾     if(!grl.hasNext()){
99 ▾       createJobMetrics(char4[l]);
100    }
101 ▾     else{
102       action.setRedirectURL(current);
103     }
104   }
105 }
106

```

Figure 113. Create metrics.

```

107 ▾ if(current.u_job_type == 'DS Pre-Processed Data'){
108     var char5 = [];
109     char5 = ["Imaging Exceptions", "Produced Natively", "EMAILSMART (Total)", "EMAILSMART SUPPRESSED",
110 "EMAILSMART REMAINDER", "EMAILSMART % REDUCTION"];
111     for(var m=0; m<char5.length; m++){
112         var grm = new GlideRecord('x_led_metrics_job_metric');
113         grm.addQuery('parent', current.getUniqueValue());
114         grm.addQuery('u_data_point', char5[m]);
115         grm.query();
116         if(!grm.hasNext()){
117             createJobMetrics(char5[m]);
118         }
119         else{
120             action.setRedirectURL(current);
121         }
122     }
123 }
124
125 ▾ function createJobMetrics(datapoint) {
126     var jm = new GlideRecord('x_led_metrics_job_metric');
127     jm.initialize();
128     jm.active = true;
129     jm.parent = current.getUniqueValue();
130     jm.u_data_point = datapoint;
131     jm.u_manually_created = true;
132     jm.insert();
133     //gs.log(datapoint + "is inserted");
134     action.setRedirectURL(current);
135 }
136 }

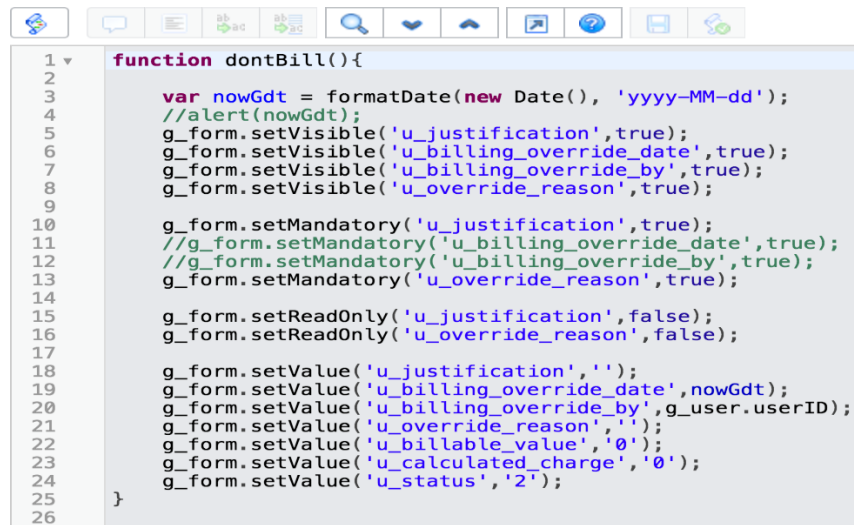
```

Figure 114. Create metrics.

6) Don't Bill

This works on Metrics table and only available to users with 'lhe_sd_admin'

role



```

1 ▾ function dontBill(){
2
3     var nowGdt = formatDate(new Date(), 'yyyy-MM-dd');
4     //alert(nowGdt);
5     g_form.setVisible('u_justification', true);
6     g_form.setVisible('u_billing_override_date', true);
7     g_form.setVisible('u_billing_override_by', true);
8     g_form.setVisible('u_override_reason', true);
9
10    g_form.setMandatory('u_justification', true);
11    //g_form.setMandatory('u_billing_override_date', true);
12    //g_form.setMandatory('u_billing_override_by', true);
13    g_form.setMandatory('u_override_reason', true);
14
15    g_form.setReadOnly('u_justification', false);
16    g_form.setReadOnly('u_override_reason', false);
17
18    g_form.setValue('u_justification', '');
19    g_form.setValue('u_billing_override_date', nowGdt);
20    g_form.setValue('u_billing_override_by', g_user.userID);
21    g_form.setValue('u_override_reason', '');
22    g_form.setValue('u_billable_value', '0');
23    g_form.setValue('u_calculated_charge', '0');
24    g_form.setValue('u_status', '2');
25 }
26

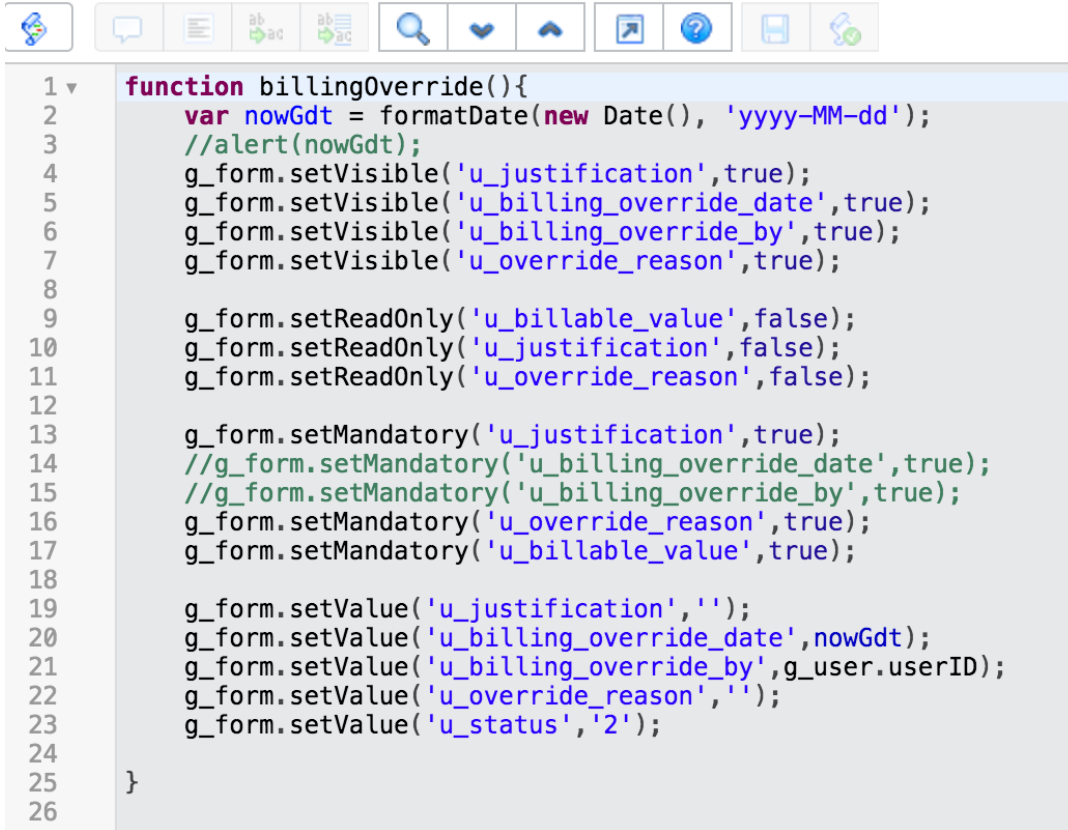
```

Figure 115. Don't bill.

It shows some hidden fields and make them mandatory.

7) Billing Override

It works almost the same way as Don't Bill.



```

1  function billingOverride(){
2      var nowGdt = formatDate(new Date(), 'yyyy-MM-dd');
3      //alert(nowGdt);
4      g_form.setVisible('u_justification',true);
5      g_form.setVisible('u_billing_override_date',true);
6      g_form.setVisible('u_billing_override_by',true);
7      g_form.setVisible('u_override_reason',true);
8
9      g_form.setReadOnly('u_billable_value',false);
10     g_form.setReadOnly('u_justification',false);
11     g_form.setReadOnly('u_override_reason',false);
12
13     g_form.setMandatory('u_justification',true);
14     //g_form.setMandatory('u_billing_override_date',true);
15     //g_form.setMandatory('u_billing_override_by',true);
16     g_form.setMandatory('u_override_reason',true);
17     g_form.setMandatory('u_billable_value',true);
18
19     g_form.setValue('u_justification','');
20     g_form.setValue('u_billing_override_date',nowGdt);
21     g_form.setValue('u_billing_override_by',g_user.userID);
22     g_form.setValue('u_override_reason','');
23     g_form.setValue('u_status','2');
24
25 }
26

```

Figure 116. Billing override.

8) Create Intake Request

This button shows on Media Notification table and is used to create an Intake Request from the current Media Notification.

Condition: `current.u_state=='Pending'`

Script:

```

1 //Client-side 'onclick' function
2 function checkPOD(){
3     if(g_form.getValue('u_pod') == ''){
4         g_form.setMandatory('u_pod', 'true');
5     }
6
7     //Call the UI Action and skip the 'onclick' function
8     gsftSubmit(null, g_form.getFormElement(), 'create_intake_req');
9 }
10
11 //Code that runs without 'onclick'
12 if(typeof window == 'undefined')
13     createIR();
14
15 function createIR(){
16     var gr = new GlideRecord('u_intake_request');
17     gr.initialize();
18     gr.u_job_type = 'Intake Request';
19     gr.u_received_from = current.u_received_from;
20     gr.u_date_media_received = current.u_received_date;
21     gr.u_incoming_tracking_num = current.u_tracking_number;
22     gr.insert();
23
24     current.u_state = 'Complete';
25     //current.u_active = false;
26     current.update();
27     action.setRedirectURL(gr);
28
29 }

```

Figure 117. Create intake request.

The next few UI actions are associated with Quality Incidents.

9) Cancel

This cancel works on Quality Incidents. It first checks with the user again to make sure the he/she really wants to cancel the incident and if so, cancels it.

This button is conditional—cancel feature only be available to Quality Reps (lhe_quality_rep role) when a member of a given Quality Rep's department reported the QINC.

lhe_quality role still need to be able to cancel any QINC regardless of any condition.

Condition	<code>(gs.hasRole('lhe_quality') (gs.hasRole('lhe_quality_rep') && current.opened_by.department == gs.getUser().getDepartmentID())) && current.state != 7</code>
Script	<pre> 1 //Client-side 'onclick' function 2 function confirmCancel(){ 3 var cancel = confirm('Are you sure you want to cancel this record?'); 4 if(cancel == true){ 5 //Call the UI Action and skip the 'onclick' function 6 gsftSubmit(null, g_form.getFormElement(), 'cancelQINC'); 7 } 8 else{ 9 return false; 10 } 11 } 12 //Code that runs without 'onclick' 13 if(typeof window == 'undefined') 14 cancelQINCIDENT(); 15 16 function cancelQINCIDENT(){ 17 current.state = 7; 18 current.update(); 19 action.setRedirectURL(current); 20 } </pre>

Figure 118. Cancel QINC.

10) Escalate

This button is used to Escalate a Quality Incident to Quality Rep Review. This can only be done from the state of User Review.

Condition	<code>(current.active == true) && (current.u_review_state == 'User Review')</code>
Script	<pre> 1 //Client-side 'onclick' function 2 function checkComments(){ 3 if(g_form.getValue('comments') == ''){ 4 g_form.setMandatory('comments', 'true'); 5 } 6 //Call the UI Action and skip the 'onclick' function 7 gsftSubmit(null, g_form.getFormElement(), 'escalate'); 8 } 9 10 //Code that runs without 'onclick' 11 if(typeof window == 'undefined') 12 setState(); 13 14 function setState(){ 15 current.u_review_state = 'Quality Rep Review'; 16 current.assigned_to = ''; 17 current.update(); 18 action.setRedirectURL(current); 19 } </pre>

Figure 119. Escalate.

11) UI actions

The following UI actions are implemented at Task level, so they can be used by various tables extending from Task. They facilitate the updating of state field.

12) In Progress

This calls the Script Include to decide the availability of the button. This is available to all tasks when state = draft/open

Condition	<code>new lhcConditionHolder().serviceDeliveryInProgress()</code>
Script	<pre> 1 setInProgress(); 2 3 function setInProgress(){ 4 5 current.assigned_to = gs.getUserID(); 6 current.state = 2; 7 current.update(); 8 action.setRedirectURL(current); 9 } 10 </pre>

Figure 120. In progress.

13) Unassign

This calls the Script Include to decide the availability of the button. This is available to all tasks with state work in progress and that are assigned to a user.

Condition	<code>new lhcConditionHolder().serviceDeliveryUnassignable()</code>
Script	<pre> 1 //Client-side 'onclick' function 2 function needCustComment(){ 3 if(g_form.getValue('comments') == ''){ 4 g_form.setMandatory('comments', 'true'); 5 } 6 //Call the UI Action and skip the 'onclick' function 7 gsftSubmit(null, g_form.getFormElement(), 'unassign'); 8 } 9 10 //Code that runs without 'onclick' 11 if(typeof window == 'undefined') 12 unassignUser(); 13 14 function unassignUser(){ 15 current.state = 1; 16 current.assigned_to = ""; 17 current.update(); 18 action.setRedirectURL(current); 19 } </pre>

Figure 121. Unassign.

14) Input Needed

This also calls the Script Include to decide the availability of the button. This is available to all tasks with state open/draft/work in progress and is used to let the PM know that the information provided is not sufficient.

Condition	<code>new lhcConditionHolder().serviceDeliveryInput();</code>
Script	<pre> 1 //Client-side 'onclick' function 2 function needComment(){ 3 if(g_form.getValue('comments') == ''){ 4 g_form.setMandatory('comments', 'true'); 5 } 6 //Call the UI Action and skip the 'onclick' function 7 gsftSubmit(null, g_form.getFormElement(), 'input_needed'); 8 } 9 10 //Code that runs without 'onclick' 11 if(typeof window == 'undefined') 12 setInputNeeded(); 13 14 function setInputNeeded(){ 15 current.state = 0; 16 current.assignment_group = current.parent.assignment_group; 17 current.assigned_to = ''; 18 current.update(); 19 20 var gr = new GlideRecord(current.parent.sys_class_name); 21 gr.addQuery('sys_id', current.parent); 22 gr.query(); 23 while(gr.next()){ 24 gr.u_job_status = 'Pending Input'; 25 gr.update(); 26 } 27 28 action.setRedirectURL(current); 29 } </pre>

Figure 122. Input needed.

15) Input Provided

This again calls the Script Include to decide the availability of the button. This is available to all tasks with state Pending Input and is used to let the assignment group know that the requested information is provided.



```

Script
1  current.state = 1;
2  var taskname = current.short_description;
3  //gs.log('IP1 taskname: ' + taskname);
4  var json = new JSONParser();
5  var groups = json.parse(gs.getProperty("sdtask.default.groups"));
6  var group = groups[taskname].toString();
7  //gs.log('IP2 group sysid: ' + group);
8  if(group == null){
9      //gs.log('IP3 group undefined');
10     //current.assignment_group = '';
11 }
12 else{
13     current.assignment_group = group;
14 }
15
16 /*if(current.sys_class_name == 'u_intake_task'){
17     current.assignment_group = getGroup('Intake Fulfill');
18     current.assigned_to = '';
19 }*/
20
21 current.update();
22
23 var grtask = new GlideRecord(current.sys_class_name);
24 grtask.addQuery('parent', current.parent);
25 grtask.addQuery('state', 0);
26 grtask.query();
27 if(!grtask.hasNext()){
28     var gr1 = new GlideRecord(current.parent.sys_class_name);
29     gr1.addQuery('sys_id', current.parent);
30     gr1.query();
31     while(gr1.next()){
32         gr1.u_job_status = 'In Progress';
33         gr1.update();
34     }
35 }
36
37 action.setRedirectURL(current);
38
39 function getGroup(groupname){
40     var gid, gr = new GlideRecord('sys_user_group');
41     gr.addQuery('name', groupname);
42     gr.query();
43     gr.next();
44     gid = gr.sys_id;
45     return gid;
46 }

```

Figure 123. Input provided.

16) Complete

This sets the task to complete and is available to any task that is not already complete, and whose Task name doesn't contain the phrase 'QC' or 'Final Check'. This is because tasks with the names QC or Final Check are closed using other UI actions—'QC Pass' and 'QC Fail'.

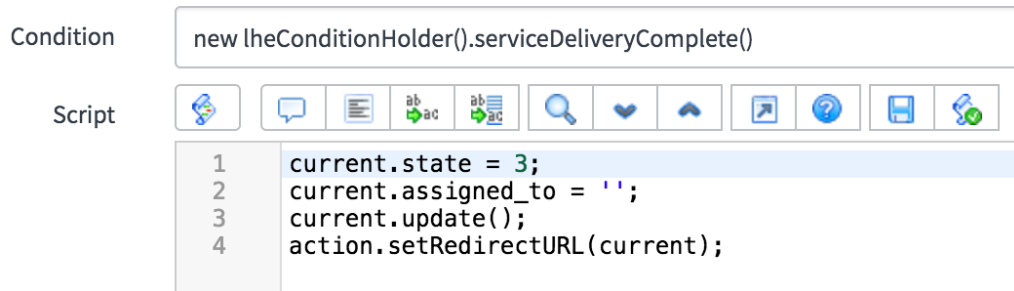


Figure 124. Complete task.

17) QC Pass

This is available to all tasks with 'QC' or 'Final Check' in its task name. This button closes the task successfully by setting the state to 'Closed Complete' or 'Complete'

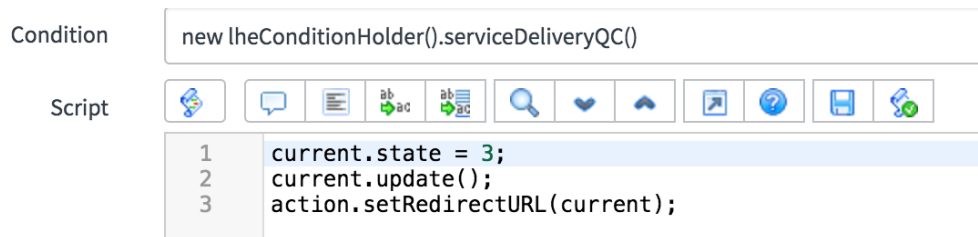


Figure 125. QC pass.

18) QC Fail

This also is available to all tasks with 'QC' or 'Final Check' in its task name. This button also closes the task but sets the state to 'Closed Incomplete' or 'Quality Fail'. This state in the task is usually used to roll back to an earlier task in the Workflows.

When a task fails QC, a quality incident is automatically generated from the failed task.

Condition	new lhcConditionHolder().serviceDeliveryQC()
Script	<pre> 1 function checkComment(){ 2 if(g_form.getValue('comments') == ''){ 3 g_form.setMandatory('comments', 'true'); 4 } 5 gsftSubmit(null, g_form.getFormElement(), 'qc_fail'); 6 } 7 if(typeof window == 'undefined') 8 setFail(); 9 function setFail(){ 10 current.state = 4; 11 current.update(); 12 //action.setRedirectURL(current); 13 var gr = new GlideRecord('u_quality_incident'); 14 gr.initialize(); 15 gr.parent = current.parent; 16 gr.u_error_reported = current.sys_id; 17 18 var user = gs.getUser(); 19 gr.u_site_reported = user.getLocation(); 20 21 var u = new GlideRecord('sys_user'); 22 u.addQuery('sys_id', gs.getUserID()); 23 u.query(); 24 while(u.next()){ 25 gr.u_dept_reported = u.department; 26 gr.u_city_reported = u.city; 27 } 28 var sysID = gr.insert(); 29 gs.addInfoMessage('Task has failed QC and a Quality Incident has been created: ' + gr.number); 30 action.setRedirectURL(gr); 31 } </pre>

Figure 126. QC fail.

19) Create Quality Incident

This is used to let the user create a Quality Incident for Service Delivery Jobs or any associated tasks.

Condition	new lhcConditionHolder().createQINC()
Script	<pre> 1 var gr = new GlideRecord('u_quality_incident'); 2 gr.initialize(); 3 if(current.parent.sys_class_name == 'u_matter' && current.sys_class_name != 'u_matter_task'){ 4 gr.parent = current.sys_id; //SD Job in which the incident is associated to 5 } 6 else{ 7 gr.parent = current.parent; 8 } 9 if(current.parent.sys_class_name != 'u_matter'){ 10 gr.u_error_reported = current.sys_id; //Task in which the incident is associated to 11 } 12 else if(current.sys_class_name == 'u_matter_task'){ 13 gr.u_error_reported = current.sys_id; //Task in which the incident is associated to 14 } 15 else{ 16 gr.u_error_reported = ''; 17 } 18 var user = gs.getUser(); 19 gr.u_site_reported = user.getLocation(); 20 21 var u = new GlideRecord('sys_user'); 22 u.addQuery('sys_id', gs.getUserID()); 23 u.query(); 24 while(u.next()){ 25 gr.u_dept_reported = u.department; 26 gr.u_city_reported = u.city; 27 } 28 var sysID = gr.insert(); 29 gs.addInfoMessage('Quality Incident ' + gr.number + ' has been created. '); 30 31 action.setRedirectURL(gr); </pre>

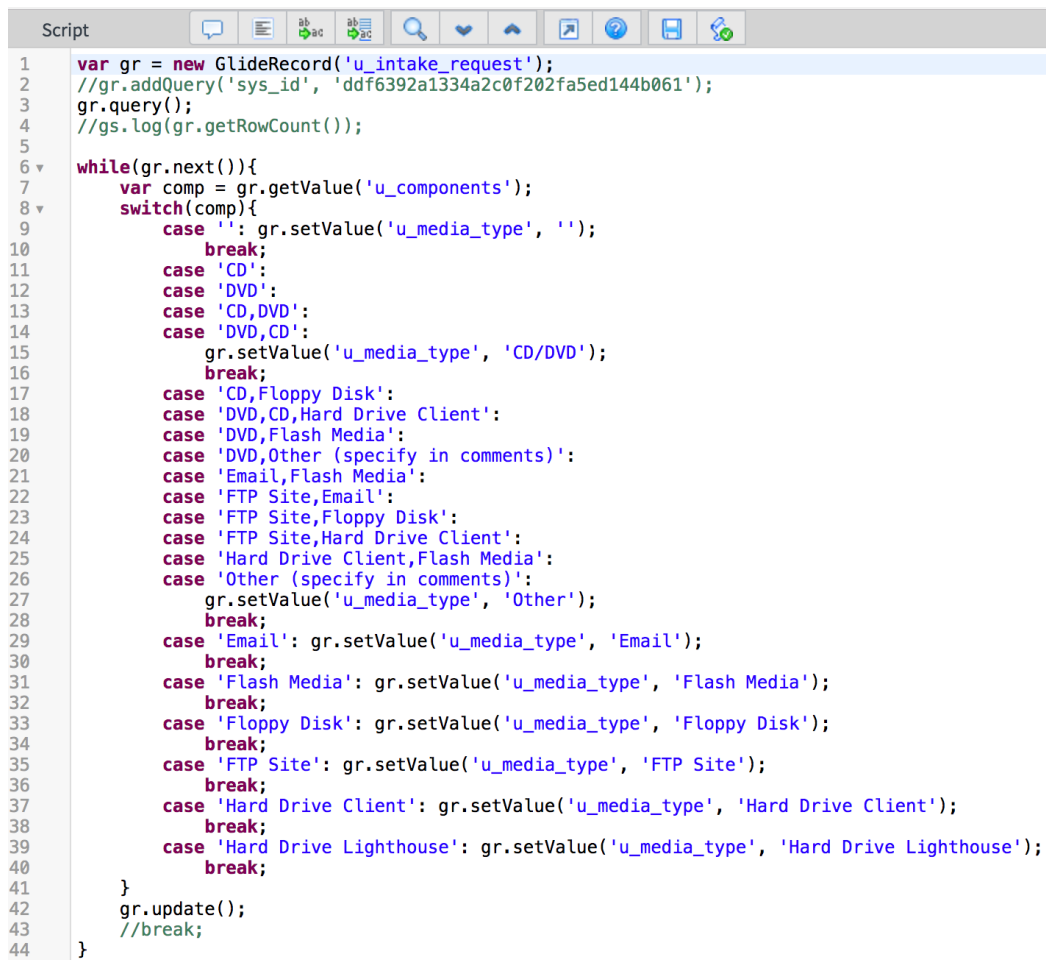
Figure 127: Create quality incident.

Fix scripts. Fix scripts are server side scripts that can be run on demand.

Usually for a one-off script, we use background scripts, but if we want to retain them to be used in other instances or to be reused in the future, fix scripts are used.

Let us look at an example.

The following script is used to update all Intake Request records to copy values from Components string field to Media Type choice field (to facilitate deleting the components field altogether).



```

Script
1  var gr = new GlideRecord('u_intake_request');
2  //gr.addQuery('sys_id', 'ddf6392a1334a2c0f202fa5ed144b061');
3  gr.query();
4  //gs.log(gr.getRowCount());
5
6  while(gr.next()){
7      var comp = gr.getValue('u_components');
8      switch(comp){
9          case '': gr.setValue('u_media_type', '');
10         break;
11         case 'CD':
12         case 'DVD':
13         case 'CD,DVD':
14         case 'DVD,CD':
15             gr.setValue('u_media_type', 'CD/DVD');
16             break;
17         case 'CD,Floppy Disk':
18         case 'DVD,CD,Hard Drive Client':
19         case 'DVD,Flash Media':
20         case 'DVD,Other (specify in comments)':
21         case 'Email,Flash Media':
22         case 'FTP Site,Email':
23         case 'FTP Site,Floppy Disk':
24         case 'FTP Site,Hard Drive Client':
25         case 'Hard Drive Client,Flash Media':
26         case 'Other (specify in comments)':
27             gr.setValue('u_media_type', 'Other');
28             break;
29         case 'Email': gr.setValue('u_media_type', 'Email');
30             break;
31         case 'Flash Media': gr.setValue('u_media_type', 'Flash Media');
32             break;
33         case 'Floppy Disk': gr.setValue('u_media_type', 'Floppy Disk');
34             break;
35         case 'FTP Site': gr.setValue('u_media_type', 'FTP Site');
36             break;
37         case 'Hard Drive Client': gr.setValue('u_media_type', 'Hard Drive Client');
38             break;
39         case 'Hard Drive Lighthouse': gr.setValue('u_media_type', 'Hard Drive Lighthouse');
40             break;
41     }
42     gr.update();
43     //break;
44 }

```

Figure 128. Copy components to media type.

In fix scripts, we need to query the tables we want to manipulate or gather data from because objects like current or previous are not available here.

Summary

In this chapter, we learned the ServiceNow tool and techniques in detail, and looked through the implemented solution so far.

Chapter V: Results, Conclusion, and Recommendations

Introduction

In this chapter, we will go through the results of the solution implemented, and what can be done to improve it in the future.

Results

Let us understand the results by answering the study questions we had at the beginning of this paper.

- How does ServiceNow work?

In the 'Literature Related to the Methodology' section in Chapter III, we gained theoretical knowledge about the ServiceNow platform. And in Chapter IV, we learnt about the tools available to us to use ServiceNow.

- How can the current system be migrated to ServiceNow without loss of data?

After implementing the tables and workflows in ServiceNow, we use Integration to integrate any third party application with ServiceNow and import data from there, which is how we imported all the Clients, Matters and Jobs from JIRA in this project.

- How can ServiceNow be used to replace JIRA, Dynamics SL and all the other media currently used?

JIRA is a ticket tracking or bug tracking system, which also provides as a tool for Agile methodology. ServiceNow also provides the same functionality with more customizable options. Dynamics SL was used for billing purposes which ServiceNow replaced, using the Metrics table and

the calculations implemented which we saw in Chapter IV. Thereby replacing both the systems all the while providing better solutions.

- How can ServiceNow be used to achieve data confidentiality, integrity, and availability?

We use ACLs and roles to restrict the data available to each role. This will guarantee data confidentiality since ACLs provide the strictest accessibility.

We have audit functionality available to every table, so if needed, any change to the data is recorded. And there's only one instance of any record, anywhere it shows up is the reference to the same record. So, any changes made to the data will be consistent. Also, no user can make changes to the data unless the ACLs allow them to. Thus, data integrity is maintained. ServiceNow is a cloud technology. Any user who has the right to access the data, can get it anytime simply by logging into the instance, thereby providing data availability.

Conclusion

This project was started to implement the most cost-effective, feasible and result-oriented solution in place of the current implementation which is error-prone, time-consuming and expensive.

Not only it replaced the current implementation brilliantly, it also has the potential to be the solution for all the other departments in the organization as well. We still haven't entirely moved over to the new solution but once it is all implemented, it is going to be very effective. People have been used to the old process for years,

which turned out to be the biggest challenge, but it only takes some training and time to make the most of it.

Future Work

ServiceNow is ever-evolving. Over the years, there have been a lot of improvements to the tool, many new features introduced every version. During the course of this project alone, we moved from Geneva version to Helsinki, and Istanbul is in the market too. We kept us apprised of the new features and made use of them in our implementation. My recommendation would be to constantly make the solutions better, by adhering to the best practices and using the new features.

In terms of the project requirements, future work will consist developing workflows for rest of the job types, developing the automation and other features that apply to the new job types, just like we did for the existing ones. Once everything has been implemented, the third party applications like JIRA and Dynamics SL can be completely cut off, and users can start using ServiceNow alone.

References

- About ServiceNow: TechWeb: Boston University.* (n.d.). Retrieved September/October 2016, from <http://www.bu.edu/tech/about/service/about/>.
- Atlassian documentation—JIRA software overview.* (n.d.). Retrieved September/October 2016, from <https://confluence.atlassian.com/jirasoftwarecloud/jira-software-overview-779293724.html>.
- JIRA tutorial: A complete guide for beginners* [Web log post]. (n.d.). Retrieved September/October 2016.
- MID server system requirements—docs.servicenow.com.* (n.d.). Retrieved September/October 2016, from https://docs.servicenow.com/bundle/helsinki-it-operations-management/page/product/mid-server/reference/r_MIDServerSystemRequirements.html.
- Microsoft dynamics SL—Synergy business solutions.* (n.d.). Retrieved September/October 2016, from <http://synergybusiness.com/microsoft-dynamics-sl/microsoft-dynamics-sl-software-solomon/>.
- Muir, M. (2014, June 30). *What ITIL, ServiceNow, and Risk & Compliance have in common.* Retrieved September/October 2016, from <http://intreis.com/compliance-til-servicenow/>.
- Request for production—FindLaw.* (n.d.). Retrieved September/October 2016, from <http://dictionary.findlaw.com/definition/request-for-production.html>.
- ServiceNow | the enterprise Cloud company.* (n.d.). Retrieved September/October 2016, from <http://www.servicenow.com/>.

ServiceNow—What is it and what are the latest trends? (2014, September 12).

Retrieved September/October 2016, from <http://www.squiresgroup.com/2014/09/12/servicenow-latest-trends/>.

The Agile movement. (2008, October 23). Retrieved September/October 2016, from <http://agilemethodology.org/>.

The basics: What is e-Discovery? (eDiscovery). (n.d.). Retrieved September/October 2016, from <http://cdslegal.com/knowledge/the-basics-what-is-e-discovery/>.

TWP. (2009, November). *Engineering firm boosts productivity management with integrated system.* Microsoft dynamics, customer solution case study.

Retrieved September/October, 2016, from http://synergybusiness.com/files/PDF/case_studies/ae/Case-Study-TWP.pdf.

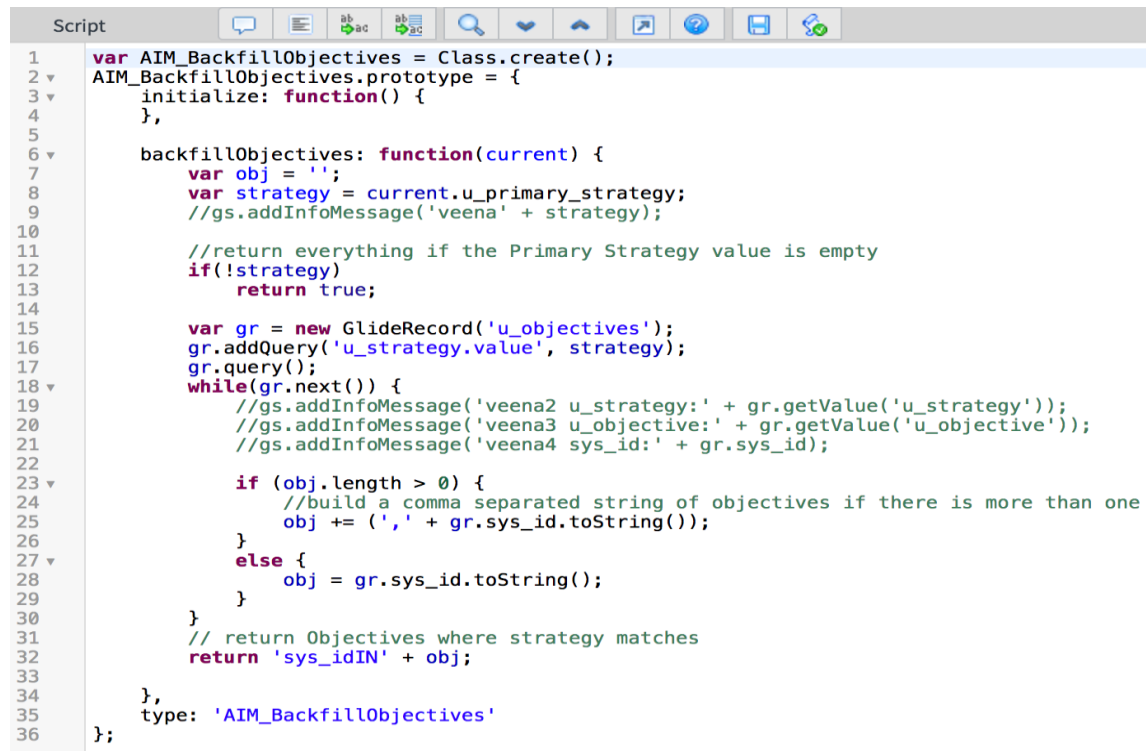
What is Agile project management?—Mountain Goat software. (n.d.). Retrieved September/October 2016, from <https://www.mountaingoatsoftware.com/agile/agile-project-management>.

What is Agile project management?—VersionOne. (n.d.). Retrieved September/October 2016, from <https://www.versionone.com/agile-project-management/>.

Appendix

Script Includes:

1) AIM_BackfillObjectives



```

1  var AIM_BackfillObjectives = Class.create();
2  AIM_BackfillObjectives.prototype = {
3  initialize: function() {
4  },
5
6  backfillObjectives: function(current) {
7  var obj = '';
8  var strategy = current.u_primary_strategy;
9  //gs.addInfoMessage('veena' + strategy);
10
11  //return everything if the Primary Strategy value is empty
12  if(!strategy)
13  return true;
14
15  var gr = new GlideRecord('u_objectives');
16  gr.addQuery('u_strategy.value', strategy);
17  gr.query();
18  while(gr.next()) {
19  //gs.addInfoMessage('veena2 u_strategy:' + gr.getValue('u_strategy'));
20  //gs.addInfoMessage('veena3 u_objective:' + gr.getValue('u_objective'));
21  //gs.addInfoMessage('veena4 sys_id:' + gr.sys_id);
22
23  if (obj.length > 0) {
24  //build a comma separated string of objectives if there is more than one
25  obj += (',' + gr.sys_id.toString());
26  }
27  else {
28  obj = gr.sys_id.toString();
29  }
30  }
31  // return Objectives where strategy matches
32  return 'sys_idIN' + obj;
33
34  },
35  type: 'AIM_BackfillObjectives'
36  };

```

Figure a: AIM_BackfillObjectives

2) AIM_calculateSDMetrics

```

Script
1  var AIM_calculateSDMetrics = Class.create();
2  AIM_calculateSDMetrics.prototype = {
3  initialize: function() {
4  },
5
6  //Initial Function that grabs the Metrics
7  getSDMetrics: function(sysId, billComponents, jStatus, matter, jobType){
8  //Set variables that will be used throughout the code
9  //gs.info("sysId - " + sysId + ", billComponents - " + billComponents + ", jStatus - " + jStatus + ", matter - " + matter + ", jobType - " + jobType);
10 var metrics = new GlideRecord('x_led_metrics_job_metric');
11 //Query the metric with parent of current sysId
12 metrics.addQuery('parent',sysId);
13 metrics.addQuery('active',true);
14 metrics.query();
15 //gs.info("RowCount for getSDMetrics - " + metrics.getRowCount());
16 while(metrics.next()){
17 //gs.info("In the while loop for metric - " + metrics.sys_id);
18 try{
19 if(metrics.u_exception != true){
20 //gs.info("Checking for datapoint - " + metrics.u_data_point + ', '+ metrics.parent.number + ', ' + metrics.u_exception + ', ' + metrics.u_value);
21 //Check For Special Cases
22 if(this.specialCase(metrics.u_data_point,sysId,metrics.u_exception,metrics.sys_id,matter,jobType,billComponents,metrics.u_value) != 0){
23 //gs.info("specialCase calculated - " + metrics.u_data_point + ", moving to next metric.");
24 continue;
25 }
26 //Check For Regular Metrics
27 else if(this.checkRegular(metrics.u_data_point,sysId,metrics.u_exception,metrics.sys_id,matter,billComponents) != 0){
28 //gs.info("checkRegular calculated - " + metrics.u_data_point + ", moving to next metric. - ");
29 continue;
30 }
31 }
32 else{
33 //gs.info("Did not match for datapoint - " + metrics.u_data_point + ', '+metrics.parent.number);
34 //gs.info("Did not match specialCase or checkRegular");
35 this.setEmpty(metrics.sys_id);
36 continue;
37 }
38 }
39 }
40 //gs.info("Checking for datapoint (exceptions) - " + metrics.u_data_point + ', ' + metrics.u_exception + ', ' + sysId);
41 //Check For Exception Metrics
42 //gs.info("Checking for datapoint - " + metrics.u_data_point + ', '+metrics.parent.number);
43 //gs.info(this.checkException(metrics.u_data_point, sysId, metrics.u_exception, metrics.sys_id, matter, billComponents) + " - returning checkException");
44 if(this.checkException(metrics.u_data_point,sysId,metrics.u_exception,metrics.sys_id,matter,billComponents) != 0){
45 //gs.info("checkException calculated - " + metrics.u_data_point + ", moving to next metric.");
46 continue;
47 }
48 }
49 else{
50 //gs.info("Did not match checkException");
51 this.setEmpty(metrics.sys_id);
52 continue;
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

Figure b: AIM_calculateSDMetrics

```

Script
1  var AIM_calculateSDMetrics = Class.create();
2  AIM_calculateSDMetrics.prototype = {
3  initialize: function() {
4  },
5
6  //Initial Function that grabs the Metrics
7  getSDMetrics: function(sysId, billComponents, jStatus, matter, jobType){
8  //Set variables that will be used throughout the code
9  //gs.info("sysId - " + sysId + ", billComponents - " + billComponents + ", jStatus - " + jStatus + ", matter - " + matter + ", jobType - " + jobType);
10 var metrics = new GlideRecord('x_led_metrics_job_metric');
11 //Query the metric with parent of current sysId
12 metrics.addQuery('parent',sysId);
13 metrics.addQuery('active',true);
14 metrics.query();
15 //gs.info("RowCount for getSDMetrics - " + metrics.getRowCount());
16 while(metrics.next()){
17 //gs.info("In the while loop for metric - " + metrics.sys_id);
18 try{
19 if(metrics.u_exception != true){
20 //gs.info("Checking for datapoint - " + metrics.u_data_point + ', '+ metrics.parent.number + ', ' + metrics.u_exception + ', ' + metrics.u_value);
21 //Check For Special Cases
22 if(this.specialCase(metrics.u_data_point,sysId,metrics.u_exception,metrics.sys_id,matter,jobType,billComponents,metrics.u_value) != 0){
23 //gs.info("specialCase calculated - " + metrics.u_data_point + ", moving to next metric.");
24 continue;
25 }
26 //Check For Regular Metrics
27 else if(this.checkRegular(metrics.u_data_point,sysId,metrics.u_exception,metrics.sys_id,matter,billComponents) != 0){
28 //gs.info("checkRegular calculated - " + metrics.u_data_point + ", moving to next metric. - ");
29 continue;
30 }
31 }
32 else{
33 //gs.info("Did not match for datapoint - " + metrics.u_data_point + ', '+metrics.parent.number);
34 //gs.info("Did not match specialCase or checkRegular");
35 this.setEmpty(metrics.sys_id);
36 continue;
37 }
38 }
39 }
40 //gs.info("Checking for datapoint (exceptions) - " + metrics.u_data_point + ', ' + metrics.u_exception + ', ' + sysId);
41 //Check For Exception Metrics
42 //gs.info("Checking for datapoint - " + metrics.u_data_point + ', '+metrics.parent.number);
43 //gs.info(this.checkException(metrics.u_data_point, sysId, metrics.u_exception, metrics.sys_id, matter, billComponents) + " - returning checkException");
44 if(this.checkException(metrics.u_data_point,sysId,metrics.u_exception,metrics.sys_id,matter,billComponents) != 0){
45 //gs.info("checkException calculated - " + metrics.u_data_point + ", moving to next metric.");
46 continue;
47 }
48 }
49 else{
50 //gs.info("Did not match checkException");
51 this.setEmpty(metrics.sys_id);
52 continue;
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

Figure c: AIM_calculateSDMetrics

```

60 * specialCase: function(dataPoint, sdJob, exception, sysId, matter, jobType, billItems, value){
61 *   var billItem, uom;
62 *   var mClient = this.getMatterClient(matter);
63 *   if(billItems.length > 1){
64 *     billItems = billItems.split(',');
65 *   }
66 *   //gs.info("Checking for specialCase - " + dataPoint + ' - ' + sdJob + ' - ' + exception + ' - ' + sysId + ' - ' + matter + ' - ' + jobType);
67 *   if (dataPoint.indexOf("OCR Page Count") > -1){
68 *     //gs.info("This is a PROC.OCR - " + matter);
69 *     billItem = 'PROC.OCR';
70 *     return this.runMetricCalc(sysId, billItem, matter, exception);
71 *   }
72 *   else if(dataPoint.indexOf("Hard Drives (over 250GB)") > -1){
73 *     //gs.info("This is an MEDIA.LG.HD - " + matter);
74 *     billItem = 'MEDIA.LG.HD';
75 *     return this.runMetricCalc(sysId, billItem, matter, exception);
76 *   }
77 *   else if(dataPoint == "DVDs"){
78 *     billItem = 'MEDIA.DVD';
79 *     return this.runMetricCalc(sysId, billItem, matter, exception);
80 *   }
81 *   else if(dataPoint.indexOf("Client Archive Request") > -1){
82 *     //gs.info("This is a HOST.ARCHIVE.FEE - " + matter);
83 *     billItem = 'HOST.ARCHIVE.FEE';
84 *     return this.runMetricCalc(sysId, billItem, matter, exception);
85 *   }
86 *   else if ((dataPoint.indexOf("Ingested GB") > -1) && (value < 2) && (jobType == "ESI Review Volume") && (billItems.indexOf("PROC-INGEST") > -1) &&
87 * (this.getSpecialBillRate(matter, "PROC.MINIMUM") != 0)){
88 *     //gs.info("This is a PROC.MINIMUM - " + matter);
89 *     billItem = 'PROC.MINIMUM';
90 *     return this.runMetricCalc(sysId, billItem, matter, exception);
91 *   }
92 *   else if((mClient == "Johnson & Johnson") || (mClient == "Microsoft Corporation")) && ((jobType == "u_production_request") && (value < 2) && (dataPoint.indexOf('Exported/Converted
93 * GB') > -1) && (billItems.indexOf("PROD-BRAND") > -1) && (billItems.indexOf("PROD-IMAGE") > -1)){
94 *     if(this.getSpecialBillRate(matter, billItem) != 0){
95 *       //gs.info("Exported/Converted GB with client - " + mClient + ', setting billItem to PROD.MINIMUM ' + sysId);
96 *       return this.runMetricCalc(sysId, billItem, matter, exception);
97 *     }
98 *     else{
99 *       billItem = 'PROD.IMAGE';
100 *       uom = 'GB';
101 *       return this.runSpecialCalc(sysId, billItem, matter, exception, uom);
102 *     }
103 *   }
104 *   else if(jobType == "u_production_request") && (value < 2) && (dataPoint.indexOf('Production Delivered GB') > -1) && (billItems.indexOf("PROD-BRAND") > -1) &&
105 * (billItems.indexOf("PROD-IMAGE") > -1) && (mClient != "Johnson & Johnson") && (mClient != "Microsoft Corporation")){
106 *     billItem = 'PROC.MINIMUM';
107 *     if(this.getSpecialBillRate(matter, billItem) != 0){
108 *       //gs.info("Production Delivered GB and client is not J&J or MSFT, setting billItem to PROD.MINIMUM ' + sysId);
109 *       return this.runMetricCalc(sysId, billItem, matter, exception);
110 *     }
111 *     else{
112 *       billItem = 'PROD.IMAGE';
113 *       uom = 'GB';
114 *       return this.runSpecialCalc(sysId, billItem, matter, exception, uom);
115 *     }
116 *   }

```

Figure d: AIM_calculateSDMetrics

```

115 *   else if((dataPoint.indexOf('Exported/Converted GB') > -1) && (mClient == "Johnson & Johnson") || (mClient == "Microsoft Corporation")) && (billItems.indexOf("PROD-IMAGE") > -1){
116 *     billItem = 'PROD.IMAGE';
117 *     uom = 'GB';
118 *     //gs.info("Exported/Converted GB with client - " + mClient + ', setting billItem to PROD.IMAGE');
119 *     return this.runSpecialCalc(sysId, billItem, matter, exception, uom);
120 *   }
121 *   else if(dataPoint.indexOf('Production Delivered GB') > -1) && (mClient != "Johnson & Johnson") && (mClient != "Microsoft Corporation") && (billItems.indexOf("PROD-IMAGE") > -1){
122 *     billItem = 'PROD.IMAGE';
123 *     uom = 'GB';
124 *     //gs.info("Production Delivered GB and client is not J&J or MSFT, setting billItem to PROD.IMAGE');
125 *     return this.runSpecialCalc(sysId, billItem, matter, exception, uom);
126 *   }
127 *   else if ((dataPoint == "Ingested GB") && (billItems.indexOf("PROC-INGEST") > -1)){
128 *     //gs.info("This is a PROC-INGEST - " + matter);
129 *     if(this.getSpecialBillRate(matter, "PROC-INGEST") != 0){
130 *       billItem = 'PROC-INGEST';
131 *       return this.runMetricCalc(sysId, billItem, matter, exception);
132 *     }
133 *     else{
134 *       //gs.info("Could not find PROC-INGEST, Checking PROC.ALLIN");
135 *       if(this.getSpecialBillRate(matter, "PROC.ALLIN") != 0){
136 *         billItem = 'PROC.ALLIN';
137 *         return this.runMetricCalc(sysId, billItem, matter, exception);
138 *       }
139 *       else{
140 *         //gs.info("Did not match specialCase for 'Ingested GB'");
141 *         return 0;
142 *       }
143 *     }
144 *   }
145 *   else{
146 *     //gs.info("Could not find a specialCase");
147 *     return 0;
148 *   }
149 * },
150 *
151 * //Check Regular Metrics
152 * checkRegular: function(dataPoint, sdJob, exception, sysId, matter, billItems){
153 *   if(billItems == ""){
154 *     //Turn billItems (Billing Components) into an Array
155 *     billItems = billItems.split(',');
156 *     //gs.info("Loop through the Bill Items on the SD Job");
157 *     //gs.info("billItems.split(',') = " + billItems);
158 *     var flag = 0;
159 *     for (var i=0, tot=billItems.length; i < tot; i++){
160 *       //gs.info("Printing out current billItems[i] = " + billItems[i]);
161 *       var item = this.normalizeBillItem(billItems[i]);
162 *       //gs.info("ran normalizeBillItem - " + item);
163 *       //gs.info("Current Bill Item in Loop - " + item + ' with dataPoint - ' + dataPoint);
164 *       var uom = this.getJobMetricChargeCalc(dataPoint, item, exception);
165 *       var rate = this.getBillingRate(item, uom, matter);
166 *       //gs.info("Printing out item - " + item + ', ' + uom + ', ' + rate + ', ' + dataPoint);
167 *       if ((uom != 0) && (rate != 0)){
168 *         flag = 1;
169 *         //gs.info("setRegularMetric - " + item + ', ' + uom + ', ' + rate + ', ' + sysId);
170 *         return this.setRegularMetric(item, uom, rate, sysId);
171 *       }

```

Figure e: AIM_calculateSDMetrics

```

115 * else if(((dataPoint.indexOf('Exported/Converted GB') > -1) && ((mClient == "Johnson & Johnson") || (mClient == "Microsoft Corporation")) && (billItems.indexOf("PROD-IMAGE") > -1)){
116 *   billItem = 'PROD.IMAGE';
117 *   uom = 'GB';
118 *   //gs.info('Exported/Converted GB with client - ' + mClient + ', setting billItem to PROD.IMAGE');
119 *   return this.runSpecialCalc(sysId,billItem,matter,exception,uom);
120 * }
121 * else if((dataPoint.indexOf('Production Delivered GB') > -1) && (mClient != "Johnson & Johnson") && (mClient != "Microsoft Corporation") && (billItems.indexOf("PROD-IMAGE") > -1)){
122 *   billItem = 'PROD.IMAGE';
123 *   uom = 'GB';
124 *   //gs.info('Production Delivered GB and client is not J&J or MSFT, setting billItem to PROD.IMAGE');
125 *   return this.runSpecialCalc(sysId,billItem,matter,exception,uom);
126 * }
127 * else if ((dataPoint == "Ingested GB") && (billItems.indexOf("PROC-INGEST") > -1)){
128 *   //gs.info('This is a PROC.INGEST - ' + matter);
129 *   if(this.getSpecialBillRate(matter,"PROC.INGEST") != 0){
130 *     billItem = 'PROC.INGEST';
131 *     return this.runMetricCalc(sysId,billItem,matter,exception);
132 *   }
133 *   else{
134 *     //gs.info('Could not find PROC.INGEST, Checking PROC.ALLIN');
135 *     if(this.getSpecialBillRate(matter,"PROC.ALLIN") != 0){
136 *       billItem = 'PROC.ALLIN';
137 *       return this.runMetricCalc(sysId,billItem,matter,exception);
138 *     }
139 *     else{
140 *       //gs.info("Did not match specialCase for 'Ingested GB'");
141 *       return 0;
142 *     }
143 *   }
144 * }
145 * else{
146 *   //gs.info("Could not find a specialCase");
147 *   return 0;
148 * }
149 * },
150 *
151 * //Check Regular Metrics
152 * checkRegular: function(dataPoint,sdJob,exception,sysId,matter,billItems){
153 *   if(billItems != ''){
154 *     //Turn billItems (Billing Components) into an Array
155 *     billItems = billItems.split(',');
156 *     //gs.info("Loop through the Bill Items on the SD Job");
157 *     //gs.info("billItems.split(',') = " + billItems);
158 *     var flag = 0;
159 *     for (var i=0, tot=billItems.length; i < tot; i++){
160 *       //gs.info("Printing out current billItems[i] - " + billItems[i]);
161 *       var item = this.normalizeBillItem(billItems[i]);
162 *       //gs.info("ran normalizeBillItem = " + item);
163 *       //gs.info('Current Bill Item in Loop - ' + item + ' with dataPoint - ' + dataPoint);
164 *       var uom = this.getJobMetricChargeCalc(dataPoint, item, exception);
165 *       var rate = this.getBillingRate(item,uom,matter);
166 *       //gs.info("Printing out item - " + item + ', ' + uom + ', ' + rate + ', ' + dataPoint);
167 *       if ((uom != 0) && (rate != 0)){
168 *         flag = 1;
169 *         //gs.info("setRegularMetric - " + item + ', ' + uom + ', ' + rate + ', ' + sysId);
170 *         return this.setRegularMetric(item,uom,rate,sysId);
171 *       }

```

Figure f: AIM_calculateSDMetrics

```

172 ▾         else{
173             //flag = 2;
174             continue;
175         }
176     }
177 ▾     if(flag == 0){
178         this.setEmpty(sysId);
179         return 0;
180     }
181 }
182 ▾ else{
183     //gs.info(sysId + " - did not have metrics to calculate")
184     return 0;
185 }
186 },
187
188 //Check Exception Metrics
189 ▾ checkException: function(dataPoint,sdJob,exception,sysId,matter,billItems){
190     if(billItems != ''){
191         //gs.info("Running checkException with dataPoint - " + dataPoint);
192         //Turn billItems (Billing Compoments) into an Array
193         var flag = 0;
194         billItems = billItems.split(',');
195         //gs.info("Loop through the Bill Items on the SD Job");
196         //gs.info("billItem.split(',') = " + billItem);
197 ▾         for (var i = 0, tot=billItems.length; i < tot; i++){
198             //gs.info("Printing out current billItem[i] - " + billItems[i]);
199 ▾             var item = this.normalizeBillItem(billItems[i]);
200             //gs.info("ran normalizeBillItem - " + item);
201             //gs.info('Current Bill Item in Loop - ' + item + ' with dataPoint - ' + dataPoint);
202             var uom = this.getJobMetricChargeCalc(dataPoint,item,exception);
203             var rate = this.getBillingRate(item,uom,matter);
204             //gs.info("Printing out item for exception - " + item + ', ' + uom + ', ' + rate);
205 ▾             if ((uom != 0) && (rate != 0)){
206                 return this.setRegularMetric(item,uom,rate,sysId);
207             }
208 ▾             else{
209                 //flag = 2;
210                 continue;
211             }
212         }
213 ▾         if(flag == 0){
214             this.setEmpty(sysId);
215             return 0;
216         }
217     }
218 ▾     else{
219         //gs.info(sysId + " - did not have metrics to calculate");
220         return 0;
221     }
222 },
223
224 //runMetricCalc sets u_billable_value = u_value
225 ▾ runMetricCalc: function(sysId,billItem,matter,exception){
226     //gs.info('Running runMetricCalc - ' + sysId + ' - ' + billItem + ' - ' + matter + ' - ' + exception);
227     //gs.info("getSpecialBillRate - " + this.getSpecialBillRate(matter,billItem));

```

Figure g: AIM_calculateSDMetrics

```

228     var rate = this.getSpecialBillRate(matter,billItem);
229     //gs.info("setSpecialtoValue - " + this.setSpecialtoValue(sysId,billItem,matter,exception,rate));
230     if(rate != 0){
231         if(this.setSpecialtoValue(sysId,billItem,matter,exception,rate) != 0){
232             //gs.info("Ran getSpecialBillRate + setSpecialtoValue");
233             return 1;
234         }
235     }
236     else{
237         return 0;
238     }
239 },
240
241 //runSpecialCalc sets u_billable_value = u_rate
242 runSpecialCalc: function(sysId,billItem,matter,exception,uom){
243     var rate = this.getBillingRate(billItem,uom,matter);
244     //gs.info("Running runSpecialCalc - " + sysId + ', ' + billItem + ', ' + matter + ', ' + uom);
245     if(rate != 0){
246         if(this.setSpecialtoRate(sysId,billItem,matter,exception,rate,uom) != 0){
247             return 1;
248         }
249     }
250     else{
251         return 0;
252     }
253 },
254
255 setRegularMetric: function(item,uom,rate,sysId){
256     //gs.info("Running setRegularMetric");
257     var gr = new GlideRecord('x_led_metrics_job_metric');
258     gr.addQuery('sys_id',sysId);
259     gr.query();
260     while(gr.next()){
261         //gs.info("setRegularMetric for - " + gr.u_data_point + ', ' + sysId);
262         gr.u_status = '1';
263         gr.u_rate = rate;
264         gr.u_billable_item_code = item;
265         gr.u_billable_unit_of_measure = uom;
266         gr.u_billable_value = Math.round(gr.u_value*100)/100;
267         return gr.update();
268     }
269 },
270
271 setSpecialtoValue: function(sysId,billItem,matter,exception,rate){
272     //gs.info("Running setSpecialtoValue - " + sysId + ' - ' + billItem + ' - ' + matter + ' - ' + exception + ' - ' + rate);
273     var gr = new GlideRecord('x_led_metrics_job_metric');
274     gr.addQuery('sys_id',sysId);
275     gr.query();
276     while(gr.next()){
277         //gs.info("setSpecialtoValue for - " + gr.u_data_point);
278         gr.u_status = '1';
279         gr.u_rate = rate;
280         gr.u_billable_item_code = billItem;
281         gr.u_billable_unit_of_measure = this.getUnitofMeasure(matter, billItem, rate);
282         if((billItem == "PROD.MINIMUM") || (billItem == "PROC.MINIMUM")){

```

Figure h: AIM_calculateSDMetrics

```

282 ▾      if((billItem == "PROD.MINIMUM") || (billItem == "PROC.MINIMUM")){
283          gr.u_billable_value = "1";
284      }
285 ▾      else{
286          gr.u_billable_value = Math.round(gr.u_value*100)/100;
287      }
288      //gs.info("In setSpecialtoValue - set these values - " + gr.u_status + ', ' + rate + ', ' + billItem);
289      return gr.update();
290  }
291 },
292
293 ▾ setSpecialtoRate: function(sysId,billItem,matter,exception,rate,uom){
294     //gs.info("Running setSpecialtoRate - " + sysId + ' - ' + billItem + ' - ' + matter + ' - ' + exception + ' - ' + rate);
295     var gr = new GlideRecord('x_led_metrics_job_metric');
296     gr.addQuery('sys_id',sysId);
297     gr.query();
298     while(gr.next()){
299         gr.u_status = '1';
300         gr.u_rate = rate;
301         gr.u_billable_item_code = billItem;
302         gr.u_billable_unit_of_measure = uom;
303         gr.u_billable_value = Math.round(gr.u_value*100)/100;
304         return gr.update();
305     }
306 },
307
308 ▾ setEmpty: function(sysId){
309     //gs.info("Running setEmpty() for metric sysId - " + sysId);
310     var gr = new GlideRecord('x_led_metrics_job_metric');
311     gr.addQuery('sys_id',sysId);
312     gr.query();
313     while(gr.next()){
314         gr.u_status = '0';
315         gr.u_rate = '0';
316         gr.u_billable_item_code = ' ';
317         gr.u_billable_unit_of_measure = ' ';
318         gr.u_billable_value = '0';
319         gr.u_calculated_charge = '0';
320         gr.update();
321     }
322 },
323
324 ▾ normalizeBillItem: function(billItem){
325     //gs.info(billItem + " - This is the billitem in normalizeBillItem");
326     if(billItem.indexOf('PROD-BRAND') != -1){
327         return 'PROD.BRAND';
328     }
329     if(billItem.indexOf('PROD-IMG2IMG') != -1){
330         return 'PROD.IMG2IMG';
331     }
332     if(billItem.indexOf('PROD-IMAGE') != -1){
333         return 'PROD.IMAGE';
334     }
335     if(billItem.indexOf('PROC-EXPORT') != -1){
336         return 'PROC.EXPORT';
337     }
338     if(billItem.indexOf('PROC-HYBRID') != -1){

```

Figure i: AIM_calculateSDMetrics


```

341 ▾   if(billItem.indexOf('PROC-OCR') != -1){
342     return 'PROC.OCR';
343   }
344 ▾   if(billItem.indexOf('PROC-INGEST') != -1){
345     return 'PROC.INGEST';
346   }
347 ▾   if(billItem.indexOf('PROC-FILTER') != -1){
348     return 'PROC.FILTER';
349   }
350 ▾   if(billItem.indexOf('PROC-PREPROC') != -1){
351     return 'PROC.PREPROC';
352   }
353 ▾   else{
354     return billItem.toString();
355   }
356 },
357
358 ▾   getJobMetricChargeCalc: function(dataPoint,billItem,exception){
359     var calc,uom;
360     //gs.info("Running getJobMetricChargeCalc with values - " + dataPoint + " , " + billItem + " , " + " , " + exception);
361     calc = new GlideRecord('u_job_metrics_charge_calculations');
362     calc.addQuery('u_data_point',dataPoint);
363     calc.addQuery('u_billing_item_code',billItem);
364     calc.addQuery('u_exception',exception);
365     calc.addQuery('u_active',true);
366     calc.query();
367 ▾   if(!calc.hasNext()){
368     //gs.info('No match of ' + billItem + ' - ' + dataPoint + ' , ' + exception);
369     return 0;
370   }
371 ▾   else{
372 ▾     while(calc.next()){
373       uom = calc.u_unit_of_measure;
374       //gs.info('getJobMetricChargeCalc returning values - ' + billItem + ' , ' + uom + ' , ' + exception);
375       return uom;
376     }
377   }
378 },
379
380 ▾   getSpecialBillRate: function(matter,billItem){
381     var billRate = new GlideRecord('u_matter_billing_rates');
382     //gs.info('Bill Rate Calc - ' + billItem);
383     billRate.addQuery('u_matter', matter);
384     billRate.addQuery('u_bill_code', billItem);
385     billRate.addQuery('u_active',true);
386     billRate.query();
387 ▾   if(!billRate.hasNext()){
388     //gs.info('Did not find billRate!');
389     return 0;
390   }
391 ▾   else{
392 ▾     while(billRate.next()){
393       //gs.info('Bill Rate Value - ' + billRate.u_rate);
394       return billRate.u_rate;
395     }
396   }

```

Figure j: AIM_calculateSDMetrics

```

399 v   getBillingRate: function(billItem,uom,matter){
400     var rate = new GlideRecord('u_matter_billing_rates');
401     rate.addQuery('u_matter',matter);
402     rate.addQuery('u_bill_code',billItem);
403     rate.addQuery('u_measure',uom);
404     rate.addQuery('u_active',true);
405     //gs.info('getBillingRate Running - ' + billItem + ' - ' + uom + ' - ');
406     rate.query();
407
408 v     if(!rate.hasNext()){
409         return 0;
410     }
411 v     else{
412 v         while(rate.next()){
413             return rate.u_rate;
414         }
415     }
416 },
417
418 v   autoCreateMetric: function(){
419     //gs.info("JLM - Starting autoCreateMetric");
420     var gr = new GlideRecord('u_matter_billing_rates');
421     gr.addEncodedQuery('u_matter.state=2^u_active=true^u_bill_codeLIKEPROD.IMAGE^u_measureLIKEPAGE');
422     gr.query();
423 v     while(gr.next()){
424         //gs.info('JLM - Querying Matter Code - ' + gr.u_matter);
425         var gr1 = new GlideRecord('u_service_delivery_job');
426         gr1.addQuery('parent',gr.u_matter);
427         gr1.addQuery('active',true);
428         gr1.addQuery('u_billable_component','CONTAINS','PROD-IMAGE');
429         gr1.query();
430 v         while(gr1.next()){
431             //gs.info('JLM - Querying SD Job - ' + gr1.number + "| sysID = " + gr1.sys_id);
432             var gr2 = new GlideRecord('x_led_metrics_job_metric');
433             gr2.addQuery('parent',gr1.sys_id);
434             gr2.addQuery('u_data_point','Page Count');
435             gr2.query();
436             //gs.info(gr2.getRowCount() + " - JLM - This is the Row Count");
437 v             if (gr2.getRowCount() < 2){
438 v                 while(gr2.next()){
439 v                     if (gr2.u_exception == ''){
440                         //gs.info('JLM - Creating New Page Count Metric for SD Job - ' + gr1.number + "| sysID = " + gr1.sys_id);
441                         gr2.initialize();
442                         //gr2.table = 'u_service_delivery_job';
443                         gr2.parent = gr1.sys_id;
444                         gr2.u_data_point = "Page Count";
445                         gr2.u_billable_item_code = "PROD.IMAGE";
446                         gr2.u_exception = true;
447                         gr2.active = true;
448                         gr2.insert();
449                     }
450 v                     else {
451                         //gs.info('JLM - SD Job - ' + gr1.number + ' already has metric created');
452                     }
453                 }
454             }

```

Figure k: AIM_calculateSDMetrics

```

456 ▾         else {
457             //gs.info('JLM - SD Job - ' + gr1.number + ' already has metric created');
458         }
459     }
460 }
461 }
462 },
463
464 ▾ getUnitofMeasure: function(matter, billItem, rate){
465     var billRate = new GlideRecord('u_matter_billing_rates');
466     billRate.addQuery('u_matter', matter);
467     billRate.addQuery('u_bill_code', billItem);
468     billRate.addQuery('u_rate', rate);
469     billRate.addQuery('u_active', true);
470     billRate.query();
471 ▾     if(!billRate.hasNext()){
472         return 0;
473     }
474 ▾     else{
475 ▾         while(billRate.next()){
476             return billRate.u_measure;
477         }
478     }
479 },
480
481 ▾ calculateMetric:function(sysID){
482     var gr = new GlideRecord('x_led_metrics_sd_metrics');
483     gr.addQuery('sys_id', sysID);
484     gr.query();
485 ▾     while(gr.next()){
486         gr.u_calculated_charge = gr.u_rate*gr.u_billable_value;
487         gr.update();
488     }
489 },
490
491 ▾ populateValue:function (){
492     var gr = new GlideRecord('x_led_metrics_job_metric');
493     gr.addQuery('u_data_point', 'Page Count');
494     gr.addQuery('u_exception', 'true');
495     gr.query();
496 ▾     while(gr.next()){
497         //gs.info('JLM - Current Metric ID - ' + gr.id);
498         gr.u_value = this.getMetricValue(gr.parent);
499         gr.update();
500     }
501 }
502 },
503

```

Figure I: AIM_calculateSDMetrics

```

504 ▾   getMetricValue: function (id){
505       var gr = new GlideRecord('x_led_metrics_job_metric');
506       gr.addQuery('u_data_point','Page Count');
507       gr.addQuery('u_exception', 'false');
508       gr.addQuery('parent',id);
509       gr.query();
510 ▾   while(gr.next()){
511       //gs.info('JLM - Current Metric ID - ' + gr.id + " with value - " + gr.u_value);
512       return gr.u_value;
513   }
514   },
515
516 ▾   getMatterClient: function (sysId){
517       //gs.info("Running getMatterClient for sysId - " + sysId);
518       var matter = new GlideRecord('u_matter');
519       matter.addQuery('sys_id',sysId);
520       matter.query();
521 ▾   while(matter.next()){
522       var companyName = new GlideRecord('core_company');
523       companyName.addQuery('sys_id',matter.company);
524       companyName.query();
525 ▾   while(companyName.next()){
526       //gs.info("Found companyName - " + companyName.getValue('name'));
527       return companyName.getValue('name');
528   }
529   }
530   },
531 };
532

```

Figure m: AIM_calculateSDMetrics

3) AIM_calculateMatterMetrics

```

Script
1   var AIM_calculateMatterMetrics = Class.create();
2   AIM_calculateMatterMetrics.prototype = {
3   initialize: function() {
4   },
5   },
6
7   calculateMetric: function(eventTrigger,matter){
8   try{
9   var dataPoint,eventArray,metrics,exception,newArray,descArray;
10  if(eventTrigger == ''){
11  //gs.info("eventTrigger is Empty");
12  newArray = new global.ArrayUtil();
13  eventArray = this.getEventDate(matter);
14  //gs.info("eventArray before .unique - " + eventArray + matter);
15  eventArray = newArray.unique(eventArray);
16  //gs.info("eventTrigger is not populated (after unique) - " + eventArray + ', ' + matter);
17  if(eventArray == ''){
18  //gs.info("eventArray is empty, continuing - " + matter);
19  return;
20  }
21  } else{
22  //gs.info("attempting to run getEventDate() - " + eventArray);
23  }
24  } else{
25  newArray = new global.ArrayUtil();
26  eventTrigger = newArray.ensureArray(eventTrigger);
27  eventArray = newArray.unique(eventTrigger);
28  //gs.info("eventTrigger is populated - " + eventArray + ', ' + matter);
29  }
30  if(eventArray != ''){
31  for(var i=0,tot=eventArray.length; i < tot; i++){
32  eventDate = eventArray[i];
33  //gs.info("Verifying buildDataPointArray - " + matter + ", " + eventDate + ", series: " + [i]);
34  descArray = this.buildDescriptionArray(matter,eventDate);
35  descArray = newArray.unique(descArray);
36  for(var j=0,tat=descArray.length; j < tat; j++){
37  //gs.info("Verifying buildDescriptionArray - " + matter + ", " + descArray[j] + ", "+ eventDate + ", series: " + [j]);
38
39
40  if(this.buildDataPointArray(matter,eventDate,descArray[j]) != 0){
41  continue;
42  }
43  } else{
44  continue;
45  }
46  }
47  }
48  } else{
49  //gs.info("eventArray is empty for " + matter);
50  return;
51  }
52  }
53  }
54  catch(err){
55  //gs.info("Error in Calculate Metric - " + err.message);
56  }

```

Figure n: AIM_calculateMatterMetrics

```

60 + checkValid: function(dataPoint,matter,eventDate,description){
61 //gs.info("Running checkValid for - " + metric_sysId + ', ' + dataPoint);
62 var gr = new GlideRecord('u_job_metrics_charge_calculations');
63 gr.addQuery('u_active',true);
64 gr.addQuery('u_data_point',dataPoint);
65 gr.query();
66 //gs.info("RowCount for checkValid - " + gr.getRowCount() + ', ' + eventDate + ', ' + dataPoint + ', ' + matter + ', ' + description);
67 if(!gr.hasNext()){
68 //gs.info("checkValid is not valid for - " + ', ' + eventDate + ', ' + dataPoint + ', ' + matter + ', ' + description);
69 return 0;
70 }
71 }
72 else{
73 while(gr.next()){
74 //gs.info("checkValid for - found values - " + gr.u_unit_of_measure + ', ' + gr.u_billing_item_code);
75 return this.checkRates(matter,dataPoint,gr.u_unit_of_measure,gr.u_billing_item_code,eventDate,description);
76 }
77 }
78 },
79
80
81 + checkRates: function(matter,dataPoint,uom,billItem,eventDate,description){
82 //gs.info("Running checkRates for - " + metric_sysId + ', ' + billItem + uom);
83 var rate = new GlideRecord('u_matter_billing_rates');
84 rate.addQuery('u_matter',matter);
85 rate.addQuery('u_bill_code',billItem);
86 rate.addQuery('u_measure',uom);
87 rate.addQuery('u_active',true);
88 rate.query();
89 //gs.info("RowCount of checkRates - " + rate.getRowCount() + ', ' + billItem + ', ' + uom + ', ' + matter + ', ' + description);
90 if(!rate.hasNext()){
91 //gs.info("checkRates did not find a rate - " + billItem + ', ' + eventDate + ', ' + dataPoint + ', ' + matter + ', ' + uom + ', ' + description);
92 return 0;
93 }
94 else{
95 while(rate.next()){
96 //gs.info("checkRates found a rate - going to setValues for " + billItem + ', ' + eventDate + ', ' + dataPoint + ', ' + matter + ', ' + uom + ', ' + description);
97 return this.setValues(dataPoint,uom,billItem,rate.u_rate,eventDate,matter,description);
98 }
99 }
100 },
101
102 + setValues: function(dataPoint,uom,billItem,rate,eventDate,matter,description){
103 var setMetric = new GlideRecord('x_led_metrics_matter_metric');
104 setMetric.addQuery('u_data_point',dataPoint);
105 setMetric.addQuery('u_event_string',eventDate);
106 setMetric.addQuery('description',description);
107 setMetric.addQuery('parent',matter);
108 setMetric.addQuery('active',true);
109 setMetric.query();
110 //gs.info("RowCount of setValues - " + setMetric.getRowCount() + ', ' + eventDate + ', ' + matter + ', ' + description);
111 while(setMetric.next()){
112 setMetric.u_billable_unit_of_measure = uom;
113 setMetric.u_billable_item_code = billItem;
114 setMetric.u_rate = rate;
115 setMetric.u_billable_value = (Math.round(setMetric.getValue('u_value')*100))/100;
116 setMetric.u_status = '1';

```

Figure o: AIM_calculateMatterMetrics

```

117 setMetric.update();
118 //gs.info("setValues setting - " + setMetric.sys_id + ', ' + eventDate + ', ' + matter + ', ' + description);
119 }
120 return this.setOtherValues(eventDate,matter,description);
121 },
122
123 + setOtherValues: function(eventDate,matter,description){
124 //gs.info("Starting setOtherValues - " + eventDate + ', ' + matter);
125 var setMetric = new GlideRecord('x_led_metrics_matter_metric');
126 setMetric.addQuery('parent',matter);
127 setMetric.addQuery('u_event_string',eventDate);
128 setMetric.addQuery('description',description);
129 setMetric.addQuery('active',true);
130 setMetric.addQuery('u_status','0');
131 setMetric.query();
132 //gs.info("RowCount of setOtherValues - " + setMetric.getRowCount() + ', ' + eventDate + ', ' + matter + ', ' + description);
133 while(setMetric.next()){
134 setMetric.active = false;
135 setMetric.u_billable_unit_of_measure = '';
136 setMetric.u_billable_item_code = '';
137 setMetric.u_rate = '';
138 setMetric.u_billable_value = '';
139 setMetric.u_status = '0';
140 setMetric.update();
141 }
142 return 1;
143 },
144
145 + getEventDate: function(matter){
146 var eventArray = [];
147 var eventD = new GlideRecord('x_led_metrics_matter_metric');
148 //gs.info("Running getEventDate() function");
149 var qc = eventD.addQuery('u_status','0');
150 qc.addOrCondition('u_status','1');
151 eventD.addQuery('active',true);
152 //eventD.addQuery('u_status','0');
153 //eventD.addOrCondition('u_status','1');
154 eventD.addQuery('parent',matter);
155 eventD.query();
156
157 //gs.info(eventD.getRowCount() + " - row count of getEventDate of " + matter);
158 while(eventD.next()){
159 //gs.info("Pushing " + eventD.u_event_string + " to eventArray" + matter);
160 eventArray.push(eventD.getValue("u_event_string"));
161 }
162 //gs.info("This is the eventArray - " + eventArray + ', ' + matter);
163 return eventArray;
164 },
165
166 + buildDescriptionArray: function(matter,eventDate){
167 var descArray = [];
168 var metric = new GlideRecord('x_led_metrics_matter_metric');
169 metric.addQuery('parent',matter);
170 metric.addQuery('u_event_string',eventDate);
171 metric.query();

```

Figure p: AIM_calculateMatterMetrics

```

172 //gs.info("RowCount of buildDescriptionArray - " + metric.getRowCount() + ', ' + eventDate + ', ' + matter);
173 while(metric.next()){
174     descArray.push(metric.getValue("description"));
175 }
176 //gs.info("buildDataPoint Array - " + descArray + ', ' + eventDate + ', ' + matter);
177 return descArray;
178 },
179
180 buildDataPointArray: function(matter,eventDate,description){
181     var dataPointArray = [];
182     var metric = new GlideRecord('x_led_metrics_matter_metric');
183     metric.addQuery('parent',matter);
184     metric.addQuery('u_event_string',eventDate);
185     metric.addQuery('description',description);
186     metric.query();
187     //gs.info("RowCount of buildDataPointArray - " + metric.getRowCount() + ', ' + eventDate + ', ' + matter);
188     while(metric.next()){
189         dataPointArray.push(metric.getValue("u_data_point"));
190     }
191     //gs.info("buildDataPoint Array - " + dataPointArray + ', ' + eventDate + ', ' + matter);
192     return this.checkDataPoint(dataPointArray,matter,eventDate,description);
193 },

```

Figure q: AIM_calculateMatterMetrics

```

195 checkDataPoint: function(dataPointArray,matter,eventDate,description){
196     //gs.info("checkDataPoint running - " + dataPointArray + ", " + matter + ", " + eventDate);
197     if((dataPointArray.indexOf("SMART.EMAIL_Doc Count - Run") > -1) && (this.checkValid("SMART.EMAIL_Doc Count - Run",matter,eventDate,description) != 0)){
198         //gs.info("checkDataPoint found SMART.EMAIL_Doc Count - Run for " + matter + ", " + eventDate);
199         return 1;
200     }
201     else if((dataPointArray.indexOf("SMART.REVIEW_Doc Count - Run") > -1) && (this.checkValid("SMART.REVIEW_Doc Count - Run",matter,eventDate,description) != 0)){
202         //gs.info("checkDataPoint found SMART.REVIEW_Doc Count - Run for " + matter + ", " + eventDate);
203         return 1;
204     }
205     else if((dataPointArray.indexOf("SMART.PrivLog_Doc Count - Run") > -1) && (this.checkValid("SMART.PrivLog_Doc Count - Run",matter,eventDate,description) != 0)){
206         //gs.info("checkDataPoint found SMART.PrivLog_Doc Count - Run for " + matter + ", " + eventDate);
207         return 1;
208     }
209     else if((dataPointArray.indexOf("SMART.PrivCat_Doc Count - Run") > -1) && (this.checkValid("SMART.PrivCat_Doc Count - Run",matter,eventDate,description) != 0)){
210         //gs.info("checkDataPoint found SMART.PrivCat_Doc Count - Run for " + matter + ", " + eventDate);
211         return 1;
212     }
213     else if((dataPointArray.indexOf("SMART.PrivLog_Doc Count - SmartSeries") > -1) && (this.checkValid("SMART.PrivLog_Doc Count - SmartSeries",matter,eventDate,description) != 0)){
214         //gs.info("checkDataPoint found SMART.PrivLog_Doc Count - SmartSeries for " + matter + ", " + eventDate);
215         return 1;
216     }
217     else if((dataPointArray.indexOf("SMART.EMAIL_Doc Count - SmartSeries") > -1) && (this.checkValid("SMART.EMAIL_Doc Count - SmartSeries",matter,eventDate,description) != 0)){
218         //gs.info("checkDataPoint found SMART.EMAIL_Doc Count - SmartSeries for " + matter + ", " + eventDate);
219         return 1;
220     }
221     else if((dataPointArray.indexOf("SMART.REVIEW_Doc Count - SmartSeries") > -1) && (this.checkValid("SMART.REVIEW_Doc Count - SmartSeries",matter,eventDate,description) != 0)){
222         //gs.info("checkDataPoint found SMART.REVIEW_Doc Count - SmartSeries for " + matter + ", " + eventDate);
223         return 1;
224     }
225     else if((dataPointArray.indexOf("SMART.PrivCat_Doc Count - SmartSeries") > -1) && (this.checkValid("SMART.PrivCat_Doc Count - SmartSeries",matter,eventDate,description) != 0)){
226         //gs.info("checkDataPoint found SMART.PrivCat_Doc Count - SmartSeries for " + matter + ", " + eventDate);
227         return 1;
228     }
229     else if((dataPointArray.indexOf("SMART.PrivLog_Native Size - SmartSeries") > -1) && (this.checkValid("SMART.PrivLog_Native Size - SmartSeries",matter,eventDate,description) != 0)){
230         //gs.info("checkDataPoint found SMART.PrivLog_Native Size - SmartSeries for " + matter + ", " + eventDate);
231         return 1;
232     }
233     else if((dataPointArray.indexOf("SMART.EMAIL_Native Size - SmartSeries") > -1) && (this.checkValid("SMART.EMAIL_Native Size - SmartSeries",matter,eventDate,description) != 0)){
234         //gs.info("checkDataPoint found SMART.EMAIL_Native Size - SmartSeries for " + matter + ", " + eventDate);
235         return 1;
236     }
237     else if((dataPointArray.indexOf("SMART.REVIEW_Native Size - SmartSeries") > -1) && (this.checkValid("SMART.REVIEW_Native Size - SmartSeries",matter,eventDate,description) != 0)){
238         //gs.info("checkDataPoint found SMART.REVIEW_Native Size - SmartSeries for " + matter + ", " + eventDate);
239         return 1;
240     }
241     else if((dataPointArray.indexOf("SMART.PrivCat_Native Size - SmartSeries") > -1) && (this.checkValid("SMART.PrivCat_Native Size - SmartSeries",matter,eventDate,description) != 0)){
242         //gs.info("checkDataPoint found SMART.PrivCat_Native Size - SmartSeries for " + matter + ", " + eventDate);
243         return 1;
244     }
245     else if((dataPointArray.indexOf("Pages Imaged For Redaction") > -1) && (this.checkValid("Pages Imaged For Redaction",matter,eventDate,description) != 0)){
246         //gs.info("checkDataPoint found Pages Imaged For Redaction " + matter + ", " + eventDate);
247         return 1;
248     }
249     else{
250         //gs.info("checkDataPoint - returning 0 - " + matter + ", " + eventDate + ", " + description);
251         return 0;

```

Figure r: AIM_calculateMatterMetrics

4) lheServiceDeliveryUtil

This class is used to provide a service delivery utility for all Service Delivery functions.

```

1  var lheServiceDeliveryUtil = Class.create();
2  lheServiceDeliveryUtil.prototype = {
3  initialize: function(currently) {
4      this.currently = currently;
5  },
6  jobCode: function(tableOrType){
7      var code;
8      code = "matter_"+"IS000";
9      return code;
10 },
11
12 transferTo: function (table,record) {
13     //Function to transfer to another task record
14     var transferRecord;
15     transferRecord = new GlideRecord(table);
16     transferRecord.initialize();
17     transferRecord.setWorkflow(false);
18     for (var field in record) {
19         if (transferRecord[field] == undefined)
20             continue;
21         if(field == "sys_class_name")
22             continue;
23         transferRecord[field] = record[field];
24     }
25
26     transferRecord.insert();
27     gs.print(table);
28     gs.print(transferRecord.number);
29 },
30
31 /*
32 This function updates the evidence reference field this will be depreciated after the JIRA integration is removed.
33
34 */
35 evidenceStringUpdate: function(evidenceArray){
36     var evidence = new GlideRecord("x_led_evidence_evidence");
37
38     //List of delimiters.
39     if(evidenceArray.indexOf(",") >= 0)
40         evidenceArray = evidenceArray.split(",");
41     else if(evidenceArray.indexOf(";") >= 0)
42         evidenceArray = evidenceArray.split(";");
43     else if(evidenceArray.indexOf(":") >= 0)
44         evidenceArray = evidenceArray.split(":");
45
46     //Check for items in the array if it exists as an actual record.
47     for(var evItem in evidenceArray){
48
49         if(evidence.get("number", evidenceArray[evItem].trim()))
50             //there must be a strict conversion to string in this case.
51             evidenceArray[evItem]= evidence.sys_id.toString();
52         else{
53             evidenceArray[evItem] = "";
54             gs.error("No Existing evidence record: " + evidenceArray[evItem], "Evidence Translation");
55         }
56     }
57     return evidenceArray.toString();
58 }

```

Figure s: lheServiceDeliveryUtil

```
58 ▾  /*Creates ESI Evidence Folder for powershell
59     */
60
61 ▾  checkEsiPowershellResult: function(powershell, esi){
62     var powerResult, condition, esiTask;
63     esiTask = new GlideRecord("u_intake_task");
64
65     condition = true;
66     gs.log(powershell);
67     powershell = powershell.replace(/\\/gi, "\\");
68     gs.log(powershell);
69     powerResult =JSON.parse(powershell);
70
71
72 ▾     if (JSUtil.nil(powerResult[0].result.success))
73 ▾         condition = condition && checkException(powerResult[0].result.fail);
74
75 ▾     var paths = powerResult[1].paths;
76
77 ▾     for (var aPath in paths) {
78
79 ▾         if (JSUtil.notNull(paths[aPath].result.success))
80             continue;
81
82 ▾         condition = condition && checkException(paths[aPath].result.fail);
83
84     }
85
86 ▾     if(condition){
87         esiTask.get(esi);
88         esiTask.state = 3;
89         esiTask.update();
90     }
91
92 ▾     function checkException(message) {
93         var result = true;
94
95         gs.info(message.indexOf("already exists") == -1);
96
97         if (message.indexOf("already exists") == -1)
98             result = false;
99
100        gs.info(result);
101        return true;
102    }
103
104
105 },
106
107
```

Figure t: lheServiceDeliveryUtil


```

108 *  setupESIFolder:function(eid){
109     var code, matter, evidence, script;
110     matter = current.parent.getRefRecord();
111     code = matter.company.u_client_code;
112     evidence = new GlideRecord("x_led_evidence_evidence");
113     evidence.get(eid);
114     evidence = evidence.number.toString().split("-")[0];
115     matter = matter.number.toString().split("-")[1];
116     script = "Import-Module D:\\ServiceNow\\ServiceNowMatterAdmin.ps1; $result = createEsiEvidence -client "+code+" -matter "+matter+" -evidence "+evidence+"; $result2 =
117     createVaultFolder -client "+code+" -matter "+matter+" -evidence "+evidence+"; write-output [$result ',' $result2];";
118     gs.addInfoMessage(script);
119     return script;
120 }
121 /*
122 runNextFlow: Initiates the next flow in the Service Delivery Job item.
123 tableException is a list of tables in which should be skipped according to the property: lhe.job.flow.skip.
124 lheWorkflowUtil is a utility class which allows for a workflow to be initiated.
125 Note: Standard- all Job Types must have one class ever.
126 */
127 runNextFlow:function(){
128     var workflow, tableException, currentTable;
129     workflow = new GlideRecord("wf_workflow");
130     currentTable = this.currently.sys_class_name.toString();
131     tableException = JSON.parse(gs.getProperty("lhe.job.flow.skip"));
132     if(tableException.indexOf(currentTable) >= 0)
133         return;
134     if(workflow.get("table", currentTable))
135         new lheWorkflowUtil(workflow.sys_id.toString(), false, {}, current);
136 }
137 etsAssignmentGroups:function(){
138     var groups, table, groupRecord, aUtil, flag;
139     aUtil = new ArrayUtil();
140     table = current.sys_class_name;
141     groups = JSON.parse(gs.getProperty("lhe.ets.groups"));
142 }

```

Figure u: lheServiceDeliveryUtil

```

164     groupRecord = new GlideRecord("sys_user_group");
165
166     flag = true;
167
168     for(var group in groups){
169         if(aUtil.indexOf(groups[group],table) > 0){
170             groups = group.toString();
171             flag = false;
172             break;
173         }
174     }
175
176 }
177
178
179 if(flag)
180     groupRecord=current.assignment_group;
181 else
182     groupRecord.get("name", groups);
183
184 return groupRecord.sys_id.toString();
185
186 },
187
188 /*
189 setMatter: accepts a parameter called matter and activeTarget.
190 Returns a matter sys_id, null if no matter exists, or ignore if the target record is false.
191
192 */
193
194 setMatter:function(matter, activeTarget, action){
195
196     var gr, response;
197
198     gs.log(activeTarget, "Service Delivery Result");
199
200     if(!activeTarget && action!="insert")
201         return "ignore";
202
203     gr = new GlideRecord('u_matter');
204     if(!gr.get('number', matter)){
205         //ignore = true;
206         response = '';
207     }
208     else{
209         response = gr.sys_id;
210     }
211
212     return response;
213 },
214
215 /*
216 Service Delivery Task assignment. Reserved for future enhancement
217
218
219 */

```

Figure v: lheServiceDeliveryUtil

```

220     assignSdTaskToGroup:function(args){
221
222         var group;
223
224         group = this.currently.assignment_group;
225
226         return group;
227     },
228
229
230     type: 'lheServiceDeliveryUtil'
231 };

```

Figure w: lheServiceDeliveryUtil

UI ACTIONS:

1) User Review

Condition	(current.active == true) && (current.u_review_state == 'Quality Rep Review' && (gs.hasRole('lhe_quality') gs.hasRole('lhe_quality_rep')))
Script	<pre> 2 function checkComments(){ 3 var answer; 4 if(g_form.getValue('comments') == '') 5 g_form.setMandatory('comments', 'true'); 6 if(g_form.getValue('assigned_to') == '') 7 g_form.setMandatory('assigned_to', 'true'); 8 if(g_form.getValue('assigned_to') != g_form.getValue('u_user')){ 9 answer = confirm('Assignee does not match selected user. Is this the desired intent?'); 10 if(answer == true){ 11 //Call the UI Action and skip the 'onclick' function 12 gsftSubmit(null, g_form.getFormElement(), 'user_review'); 13 } 14 else{ 15 return false; 16 } 17 } 18 else{ 19 //Call the UI Action and skip the 'onclick' function 20 gsftSubmit(null, g_form.getFormElement(), 'user_review'); 21 } 22 } 23 24 //Code that runs without 'onclick' 25 if(typeof window == 'undefined') 26 setState(); 27 28 function setState(){ 29 current.u_review_state = 'User Review'; 30 current.update(); 31 action.setRedirectURL(current); 32 } </pre>

Figure x: User Review

2) Ready

Condition	(current.active == true) && (current.u_review_state == 'Pending')
Script	<pre> 1 if(current.u_request_delivered == 'Yes' current.u_critical == 'Yes'){ 2 current.u_review_state = 'Quality Rep Review'; 3 current.assigned_to = ''; 4 } 5 else{ 6 if(current.u_error_created == ''){ 7 current.u_review_state = 'Quality Rep Review'; 8 current.assigned_to = ''; 9 } 10 else{ 11 current.u_review_state = 'User Review'; 12 current.assigned_to = current.u_error_created.closed_by; 13 } 14 } 15 current.u_user = current.u_error_created.closed_by; 16 current.u_site_originated = current.u_user.location; 17 current.u_city_originated = current.u_user.city; 18 current.u_dept_originated = current.u_user.department; 19 20 current.update(); 21 action.setRedirectURL(current); 22 </pre>

Figure y: Ready

3) Complete

This Complete is available when the current QINC is in User review.

Condition `(current.u_review_state == 'User Review' && (gs.hasRole('lhe_quality') || gs.hasRole('lhe_quality_rep') || ((gs.hasRole('lhe_sd') && current.u_user == gs.getUserID()))))`

Script

```

1 //Client-side 'onclick' function
2 function checkQIFields(){
3
4     g_form.setMandatory('u_error_created', 'true');
5     g_form.setMandatory('u_resolution_type', 'true');
6     g_form.setMandatory('u_dept_originated', 'true');
7     g_form.setMandatory('u_city_originated', 'true');
8
9     if(g_form.getValue('u_source') == 'User Error'){
10        g_form.setMandatory('u_user', 'true');
11    }
12    //Call the UI Action and skip the 'onclick' function.
13    gsftSubmit(null, g_form.getFormElement(), 'completeQI');
14 }
15
16 //Code that runs without 'onclick'
17 if(typeof window == 'undefined')
18     setState();
19
20 function setState(){
21     current.u_review_state = 'Complete';
22     current.assigned_to = '';
23     current.update();
24     action.setRedirectURL(current);
25 }

```

Figure z: Complete QINC