

SERVICE-NOW ↔ CA AGILE CENTRAL (RALLY)

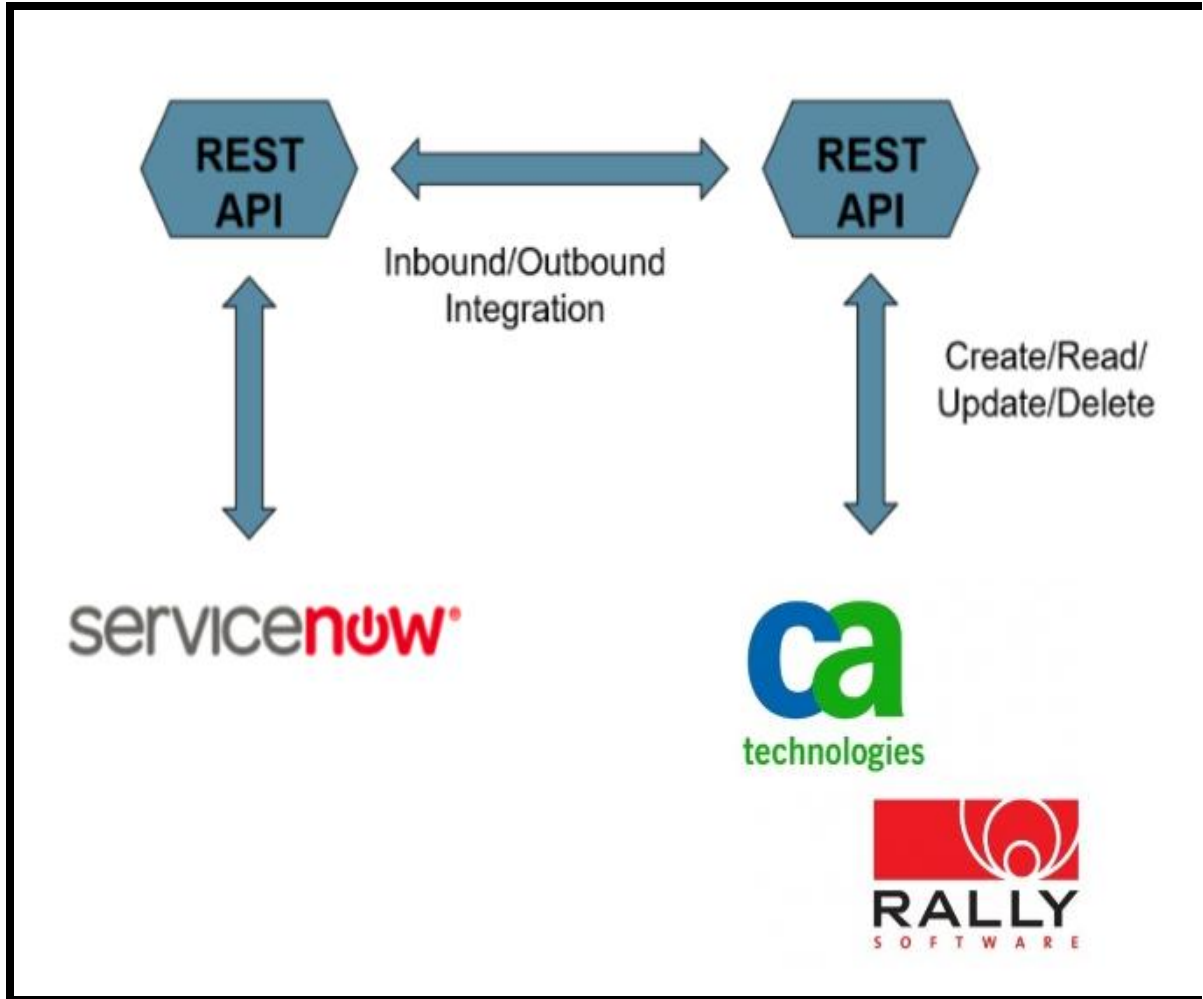
INTEGRATION DOCUMENT

Sriram Nandiraju (Ram) *

Introduction: The objective is to allow the creation of the User Story from a SNOW RITM to be automated so that transferring the data is not a manual and time-consuming process. This also will allow users to receive updates more frequently as updates to the User Story occur as the work notes within SNOW will reflect the same.

Below is the high-level architecture of the integration between Service-Now and Rally.

* Principal Software Developer in Sabre Corporation. He has 14 years of experience in IT and has worked in multiple technologies.



This document is divided into 6 sections.

1. Rally Configuration (click [here](#))
2. Service-Now configuration (click [here](#))
3. Demo (click [here](#))
4. Known Issues (click [here](#))
5. Scope for enhancement (click [here](#))
6. Research Items (click [here](#))

Rally Configuration

Rally Webhooks: They allow you to create rules in CA Agile Central that send information to an external server when the rule triggers.

Some useful applications of webhooks include:

- Receive an update when a single artifact changes
- Receive updates when any artifact changes in your subscription

Rally Webhooks Authentication: All requests to the webhooks API must be authenticated, proving that the request comes from a user with a valid and active CA Agile Central account.

Create An API Key, which you can create [here](#).

API Keys

An API Key is similar to a password, and should be protected as such. Do not share API Keys, store them in source code, or send them over insecure channels. An API Key is useful for simple interactions with the CA Agile Central API, such as a connector that is not a web application.

If you think an API Key has been compromised, either reset the API Key or delete it.

[Create New API Key](#)

My Keys

Create API Key

Description

Enter a name or a brief description indicating what the key will be used for.

Grants

ALM WSAPI Read-only

ALM will only allow GET requests from WSAPI

Full Access

Full Access to all Rally applications

API Keys

An API Key is similar to a password, and should be protected as such. Do not share API Keys, store them in source code, or send them over insecure channels. An API Key is useful for simple interactions with the CA Agile Central API, such as a connector that is not a web application.

If you think an API Key has been compromised, either reset the API Key or delete it.

[Create New API Key](#)

My Keys

Sabre DEV Snow



full-access
08/15/2017

The ZSESSIONID cookie returned when you log in to CA Agile Central.

Then, when making requests to the webhooks API, pass the key as the ZSESSIONID cookie.

The method for doing this will depend on the programming language and HTTP library that you use. Here is an example of a raw HTTP request, showing how to authenticate.

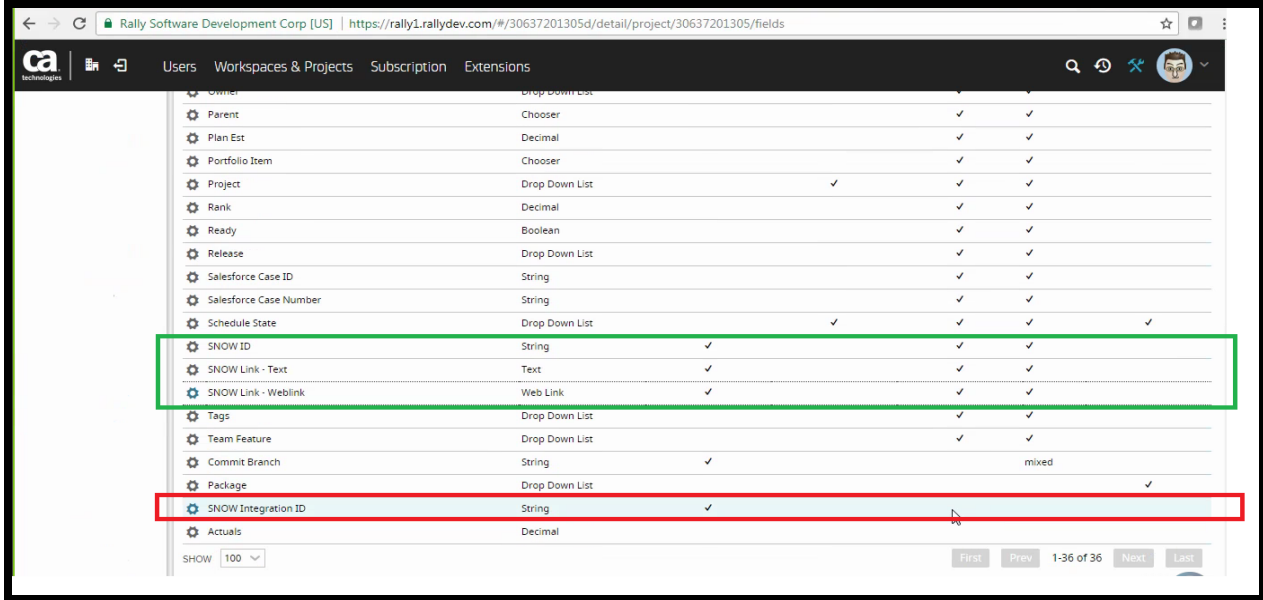
```
POST https://rally1.rallydev.com/apps/pigeon/api/v2/webhook
HTTP/1.1
Host: rally1.rallydev.com
Cookie: ZSESSIONID=blahblahblah
```

Custom Attributes in Rally: The below custom fields/attributes should be created in Rally.

c_SNOWID	
Required	false
Type	string
Max Length	256
Sortable	true
Explicit Fetch	false
Query Expression Operators	=, !=, >, <, >=, <=, contains, !contains
c_SNOWIntegrationID	
Required	false
Type	string
Max Length	256
Sortable	true
Explicit Fetch	false
Query Expression Operators	=, !=, >, <, >=, <=, contains, !contains

c_SNOWLinkText	
Required	false
Type	string
Max Length	32,768
Sortable	true
Explicit Fetch	false
Query Expression Operators	contains, !contains

c_SNOWLinkWeblink	
Required	false
One-To-One Relationship	WebLink, which has a LinkID field and a DisplayString field.
Sortable	true
Explicit Fetch	false
Query Expression Operators	This attribute cannot be used in queries.



Field/Attribute	Type	✓	✓	✓	✓
Parent	Chooser				
Plan Est	Decimal				
Portfolio Item	Chooser				
Project	Drop Down List	✓			
Rank	Decimal				
Ready	Boolean				
Release	Drop Down List				
Salesforce Case ID	String				
Salesforce Case Number	String				
Schedule State	Drop Down List		✓		
SNOW ID	String	✓			
SNOW Link - Text	Text	✓			
SNOW Link - Weblink	Web Link	✓			
Tags	Drop Down List				
Team Feature	Drop Down List				
Commit Branch	String	✓			mixed
Package	Drop Down List				✓
SNOW Integration ID	String	✓			
Actuals	Decimal				

Configuring the Rally Webhook: In order to configure the webhook, we need to know the

- fields/attributes on which the webhook gets triggered
- projects which it should monitor

In this exercise, we will configure the webhook on the User Stories in Rally. Hence go to the URL <https://rally1.rallydev.com/slm/webservice/v2.0/TypeDefinition?fetch=Attributes>

Search for "_refObjectName": "Hierarchical Requirement"

```
{
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref":
  "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/30637201358",
  "_refObjectUUID": "b973c120-9226-4fab-a4ff-4cf92514339a",
  "_objectVersion": "1",
  "_refObjectName": "Hierarchical Requirement",
  "Attributes": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref":
    "https://rally1.rallydev.com/slm/webservice/v2.0/TypeDefinition/30637201358/Attributes",
    "_type": "AttributeDefinition",
    "Count": 72
  },
  "_type": "TypeDefinition"
}
```

NOTE: The data in the highlighted URL is shown in paginated size of 20.

<https://rally1.rallydev.com/slm/webservice/v2.0/TypeDefinition/30637201358/Attributes?start=60>

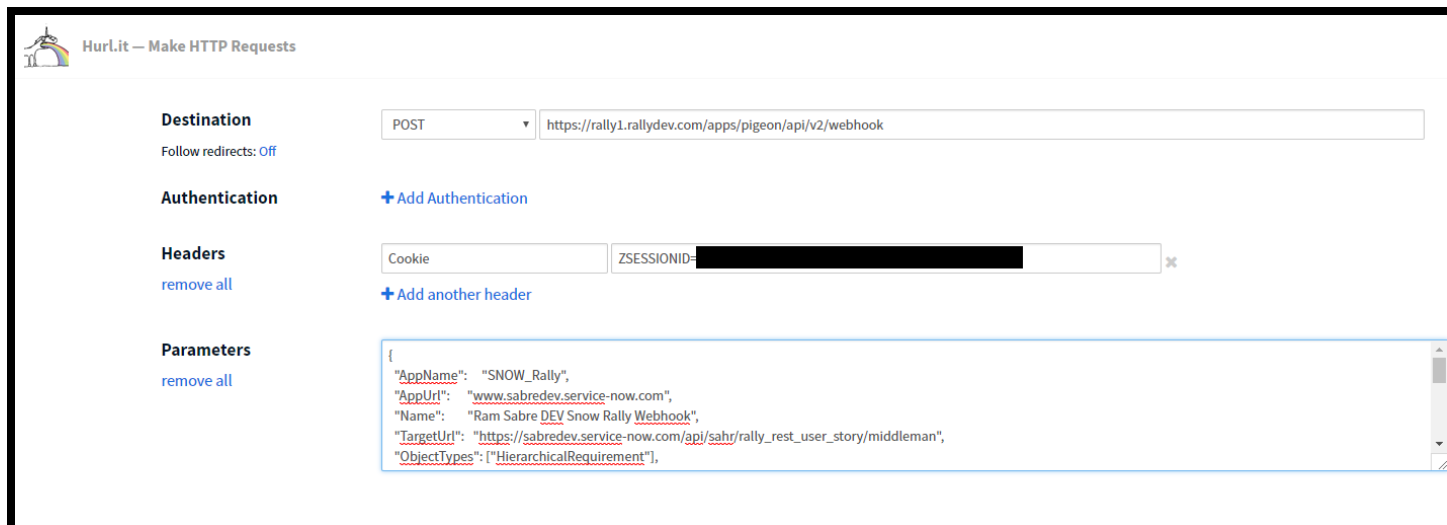

```
{
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/attributedefinition/142854949796",
  "_refObjectUUID": "e172ce22-9ec8-4b65-9595-08eb1665b90c",
  "_objectVersion": "1",
  "_refObjectName": "SNOW Integration ID",
  "CreationDate": "2017-08-10T14:41:34.658Z",
  "_CreatedAt": "Aug 10",
  "ObjectID": 142854949796,
  "ObjectUUID": "e172ce22-9ec8-4b65-9595-08eb1665b90c",
  "VersionId": "1",
  "Subscription": {},
  "Workspace": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/workspace/30637201215",
    "_refObjectUUID": "ee5948bb-365c-4869-bcbf-4c5237e4a3e4",
    "_refObjectName": "Integration Workspace",
    "_type": "Workspace"
  },
  "AllowedQueryOperators": {},
  "AllowedValueType": null,
  "AllowedValues": {},
  "AttributeType": "STRING",
  "Constrained": false,
  "Custom": true,
  "DetailedType": null,
  "ElementName": "c_SNOWIntegrationID",
  "Filterable": true,
  "Hidden": true,
  "Hideable": true,
  "MaxFractionalDigits": 0,
  "MaxLength": 256,
  "Name": "SNOW Integration ID",
  "Note": null,
  "Owned": true,
  "ReadOnly": false,
  "RealAttributeType": "STRING",
  "Required": false,
  "Sortable": true,
  "SystemRequired": false,
  "Type": "string",
  "TypeDefinition": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/30637201358",
    "_refObjectUUID": "b973c120-9226-4fab-a4ff-4cf92514339a",
    "_refObjectName": "Hierarchical Requirement",
    "_type": "TypeDefinition"
  },
  "VisibleOnlyToAdmins": false,
  "_type": "AttributeDefinition"
}
```

We want to configure this webhook on the SNOW Dawgs project

<https://rally1.rallydev.com/slm/webService/v2.0/project?workspace=https://rally1.rallydev.com/slm/webService/v2.0/workspace/30637201215&query=&start=1&pagesize=70>

```
{
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webService/v2.0/project/140242043552",
  "_refObjectUUID": "d9172927-d540-4885-9cc9-c275b3c48aa8",
  "_refObjectName": "SNOW Dawgs",
  "_type": "Project"
},
```

Rally Webhook creation: Below is the Rest web service request for creating the Rally Webhook.



The screenshot shows the Hurl.it interface for configuring an HTTP request. The destination is set to POST at `https://rally1.rallydev.com/apps/pigeon/api/v2/webhook`. The authentication is set to Cookie with the header `ZSESSIONID=` followed by a redacted value. The parameters are a JSON object:

```
{
  "AppName": "SNOW_Rally",
  "AppUrl": "www.sabredev.service-now.com",
  "Name": "Ram Sabre DEV Snow Rally Webhook",
  "TargetUrl": "https://sabredev.service-now.com/api/sahr/rally_rest_user_story/middleman",
  "ObjectTypes": ["HierarchicalRequirement"]
}
```

The below webhook definition is going to look at any **user story** in **Snow Dawgs** project which **has a non-null** value in the field/attribute **SNOW Integration ID**

```
{
  "AppName": "SNOW_Rally",
  "AppUrl": "www.sabredev.service-now.com",
  "Name": "Ram Sabre DEV Snow Rally Webhook",
  "TargetUrl": "https://sabredev.service-now.com/api/sahr/rally_rest_user_story/middleman",
  "ObjectTypes": ["HierarchicalRequirement"],
  "Expressions": [
    {
      "AttributeID": "e172ce22-9ec8-4b65-9595-08eb1665b90c",
      "Operator": "has"
    },
    {
      "AttributeName": "project",

```

```
"Operator": "=",
"Value": "d9172927-d540-4885-9cc9-c275b3c48aa8"
}
]
}
```

Emphasis was given to focus on using an Attribute's ID instead of its Name. The ID is guaranteed to match using only the intended Attribute, whereas the Name is less precise. If, for example, a new Attribute named "SNOW Integration ID" were added to another object in the model, the webhook could begin to match changes that you did not intend.

Also, using AttributeID guarantees that expressions will continue to work correctly even if an attribute get renamed, as can happen for custom attributes.

You can find Attribute IDs by examining TypeDefinitions and AttributeDefinitions defined in the WSAPI Object Model.

Below is the response from Rally after the successful creation of the webhook.

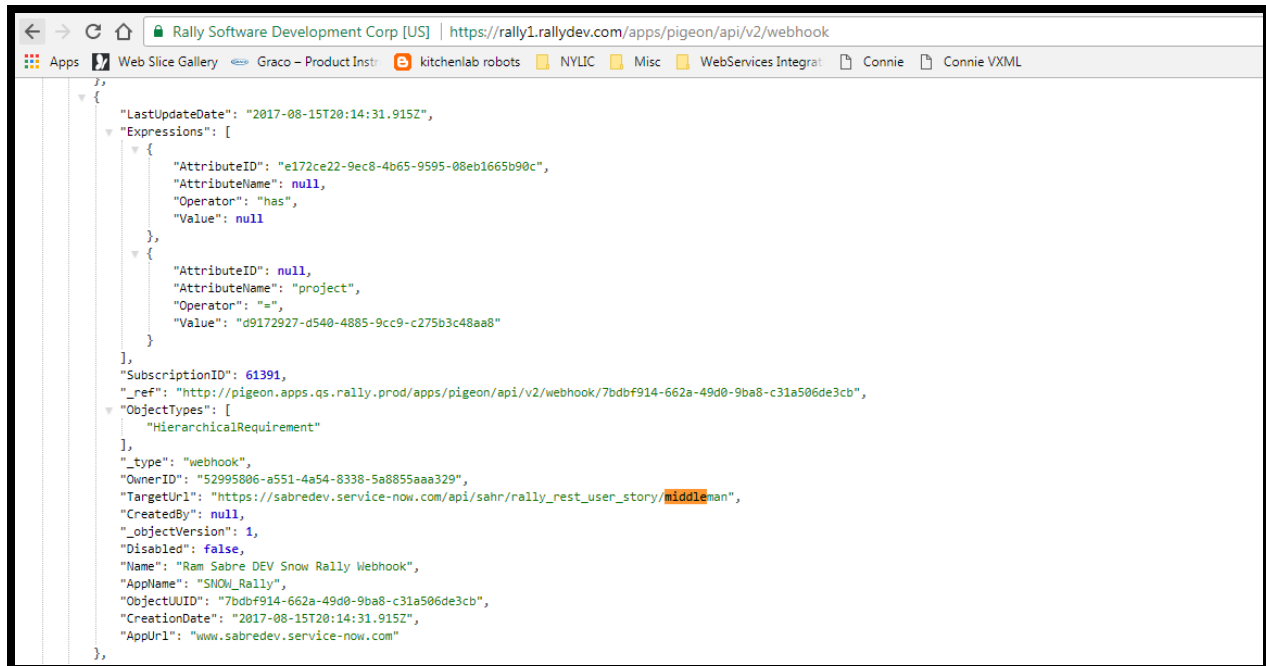
```
HEADERS
Cache-Control: no-store,max-age=0,must-revalidate
Cf-Ray: 38eed218be0f2432-IAD
Connection: keep-alive
Content-Encoding: gzip
Content-Type: application/json
Date: Tue, 15 Aug 2017 20:14:31 GMT
Server: cloudflare-nginx
Service: pigeon
Set-Cookie: __cfduid=blahblahblah; expires=Wed, 15-Aug-18 20:14:31 GMT; path=/; domain=.rallydev.com; HttpOnly
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload;
Transfer-Encoding: chunked
BODY view raw

{
  "LastUpdateDate": "2017-08-15T20:14:31.915Z",
  "Expressions": [
    {
      "AttributeID": "e172ce22-9ec8-4b65-9595-08eb1665b90c",
      "AttributeName": null,
      "Operator": "has",
      "Value": null
    },
    {
      "AttributeID": null,
      "AttributeName": "project",
      "Operator": "=",
      "Value": "d9172927-d540-4885-9cc9-c275b3c48aa8"
    }
  ],
  "SubscriptionID": 61391,
  "_ref": "http://pigeon.apps.qs.rally.prod/apps/pigeon/api/v2/webhook/7bdbf914-662a-49d0-9ba8-c31a506de3cb",
  "ObjectTypes": [
    "HierarchicalRequirement"
  ],
  "_type": "webhook",
  "OwnerID": "52995806-a551-4a54-8338-5a8855aaa329",
  "TargetUrl": "https://sabredev.service-now.com/api/sahr/rally_rest_user_story/middleman",
  "CreatedBy": null,
```

```
"_objectVersion": 1,  
"Disabled": false,  
"Name": "Ram Sabre DEV Snow Rally Webhook",  
"AppName": "SNOW_Rally",  
"ObjectUUID": "7bdf914-662a-49d0-9ba8-c31a506de3cb",  
"CreationDate": "2017-08-15T20:14:31.915Z",  
"AppUrl": "www.sabredev.service-now.com"  
}
```

We can query for the list of all the configured webhooks in Rally by hitting the URL <https://rally1.rallydev.com/apps/pigeon/api/v2/webhook>

The ObjectUUID is listed below with 7bdbf914-662a-49d0-9ba8-c31a506de3cb which is what we have received in the Rally web service response post the successful creation of the Rally Webhook.



```
},
{
  "LastUpdateDate": "2017-08-15T20:14:31.915Z",
  "Expressions": [
    {
      "AttributeID": "e172ce22-9ec8-4b65-9595-08eb1665b90c",
      "AttributeName": null,
      "Operator": "has",
      "Value": null
    },
    {
      "AttributeID": null,
      "AttributeName": "project",
      "Operator": "=",
      "Value": "d9172927-d540-4885-9cc9-c275b3c48aa8"
    }
  ],
  "SubscriptionID": 61391,
  "_ref": "http://pigeon.apps.qs.rally.prod/apps/pigeon/api/v2/webhook/7bdbf914-662a-49d0-9ba8-c31a506de3cb",
  "ObjectTypes": [
    "HierarchicalRequirement"
  ],
  "_type": "webhook",
  "OwnerID": "52995806-a551-4a54-8338-5a8855aaa329",
  "TargetUrl": "https://sabredev.service-now.com/api/sahr/rally_rest_user_story/middleman",
  "CreatedBy": null,
  "_objectVersion": 1,
  "Disabled": false,
  "Name": "Ram Sabre DEV Snow Rally Webhook",
  "AppName": "SNOW_Rally",
  "ObjectUUID": "7bdbf914-662a-49d0-9ba8-c31a506de3cb",
  "CreationDate": "2017-08-15T20:14:31.915Z",
  "AppUrl": "www.sabredev.service-now.com"
},
}
```

Service-Now Configuration

The following configuration needs to be done within Service-now

1. [Create](#) USERID with SOAP user rights for Adding attachments into Service-Now
2. [UI Action](#) – Create User Story: This UI action calls the UI page and passes variables from the RITM
3. [UI Page](#) – rally_user_story: This contains all the script in the processing script section that makes the calls to Rally through REST for creating a user story
4. [Style Sheet](#) – Rally Integration Style: Contains the style for the UI Page
5. [Scripted REST API](#) – Rally REST API: This updates SNOW RITM based on information changing in Rally

User account details

Name ID: (RALLY_REST)

* User ID: Active:

Employee number: VIP:

Title: Internal Integration User:

Default Assignment Group: Web service access only:

Password: Photo: [Click to add...](#)

Last login time:

Location and contact notification details

Notification: Location:

Calendar integration: City:

Email: Building:

Business phone: Cube location:

Secondary phone: Language:

Organization

Team: Cost center:

Department: Manager:

Company: LDAP server:

Locked out:

[Update](#) [Save](#) [Show on org chart](#)

Related Links

[Personalize all](#)

[Add to Update Set](#)

[Notification Preferences](#)

[Show Table in Schema Map](#)

[View Subscriptions](#)

[Reset a password](#)

Notification Devices (1) Requested approvals SNOW Group Memberships Groups - AD & SNOW **User's roles (31)** Assets Related CI's Delegates Subscriptions Employee Accounts Attachments(1) User Preferences

User Presence Live Users Recent audit history Audit Records (2)

User's roles [Edit...](#) for text

User = (RALLY_REST)

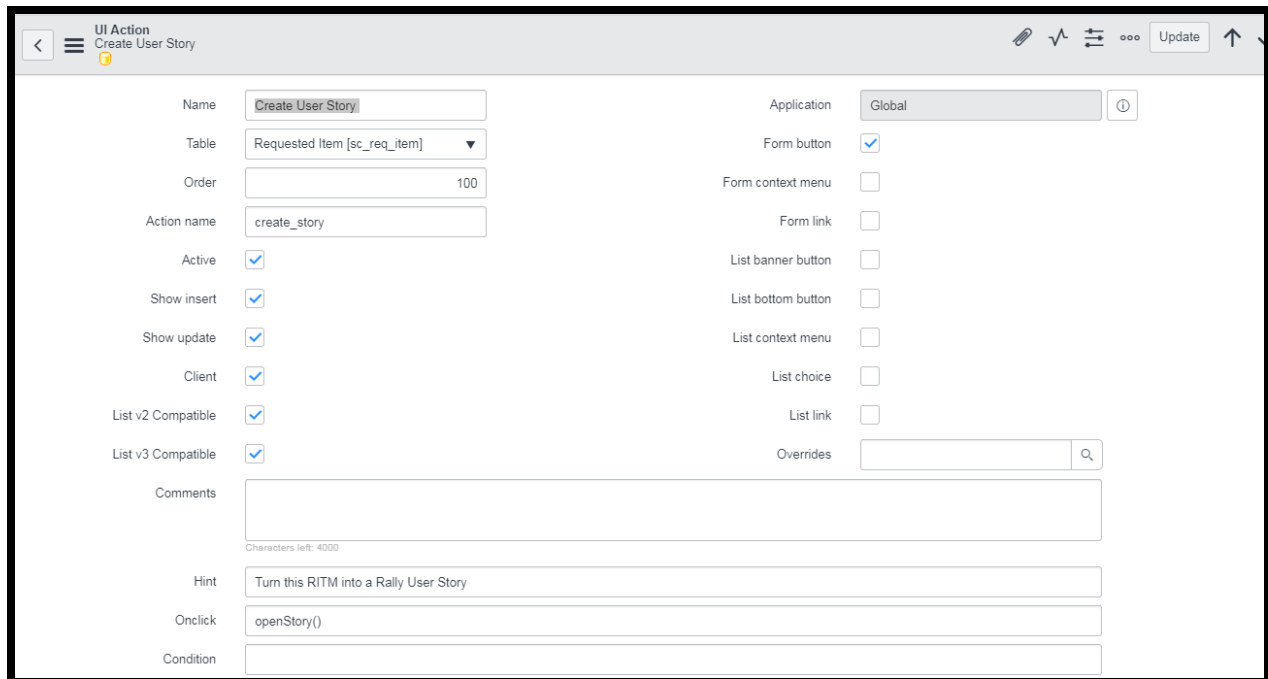
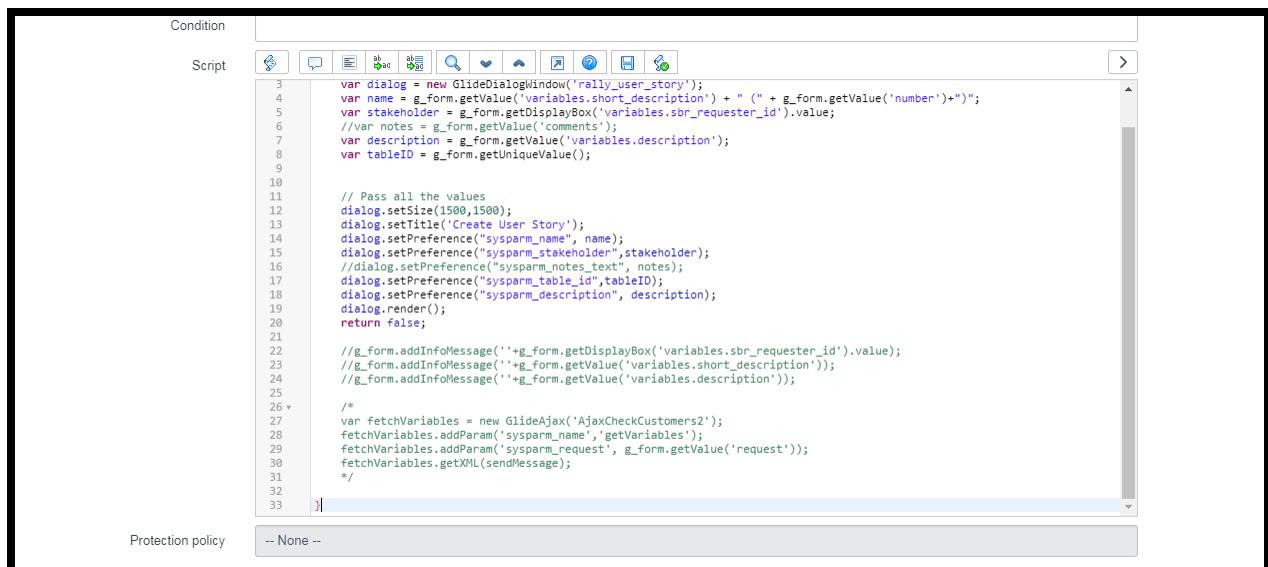
<input type="checkbox"/>	<input type="text" value="Role"/>	<input type="text" value="State"/>	<input type="text" value="Granted by"/>	<input type="text" value="Inherited"/>	<input type="text" value="Updated by"/>	<input type="text" value="Updated"/>
<input type="checkbox"/>	soap_query_update	Active	false	SG0227472	08-17-2017 15:17:03	
<input type="checkbox"/>	soap_create	Active	false	SG0227472	08-17-2017 15:17:03	
<input type="checkbox"/>	soap_create	Active	true	SG0227472	08-17-2017 15:17:03	
<input type="checkbox"/>	soap_update	Active	true	SG0227472	08-17-2017 15:17:03	
<input type="checkbox"/>	soap_script	Active	false	SG0227472	08-17-2017 15:17:03	

User (RALLY_REST)

<input type="checkbox"/>	soap_query	Active	true	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_ecc	Active	false	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_script	Active	true	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_query	Active	true	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_ecc	Active	true	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_delete	Active	true	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_update	Active	false	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap	Active	false	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	soap_delete	Active	false	SG0227472	08-17-2017 15:17:03
<input type="checkbox"/>	pa_viewer	Active	true	SG0300767	08-04-2017 15:35:29
<input type="checkbox"/>	itj	Active	false	SG0300767	08-04-2017 15:35:29
<input type="checkbox"/>	u_wat_user	Active	true	SG0300767	08-04-2017 15:35:29
<input type="checkbox"/>	gauge_maker	Active	true	SG0300767	08-04-2017 15:35:29

20 rows per page 1 to 20 of 31

UI Action – Create User Story: This UI action calls the ui page and passes variables from the RITM

```

3  var dialog = new GlideDialogWindow('rally_user_story');
4  var name = g_form.getValue('variables.short_description') + " (" + g_form.getValue('number')+");"
5  var stakeholder = g_form.getDisplayBox('variables.sbr_requester_id').value;
6  //var notes = g_form.getValue('comments');
7  var description = g_form.getValue('variables.description');
8  var tableID = g_form.getUniqueValue();
9
10
11 // Pass all the values
12 dialog.setSize(1500,1500);
13 dialog.setTitle('Create User Story');
14 dialog.setPreference("sysparm_name", name);
15 dialog.setPreference("sysparm_stakeholder", stakeholder);
16 //dialog.setPreference("sysparm_notes_text", notes);
17 dialog.setPreference("sysparm_table_id", tableID);
18 dialog.setPreference("sysparm_description", description);
19 dialog.render();
20 return false;
21
22 //g_form.addInfoMessage(''+g_form.getDisplayBox('variables.sbr_requester_id').value);
23 //g_form.addInfoMessage(''+g_form.getValue('variables.short_description'));
24 //g_form.addInfoMessage(''+g_form.getValue('variables.description'));
25
26 /*
27 var fetchVariables = new GlideAjax('AjaxCheckCustomers2');
28 fetchVariables.addParam('sysparm_name', 'getVariables');
29 fetchVariables.addParam('sysparm_request', g_form.getValue('request'));
30 fetchVariables.getXML(sendMessage);
31 */
32
33

```

```

function openStory() {
    // Get all the values that need to be passed to the ui page - rally_user_story
    var dialog = new GlideDialogWindow('rally_user_story');
    var name = g_form.getValue('variables.short_description') + " (" + g_form.getValue('number')+");"
    var stakeholder = g_form.getDisplayBox('variables.sbr_requester_id').value;
    //var notes = g_form.getValue('comments');
    var description = g_form.getValue('variables.description');
    var tableID = g_form.getUniqueValue();

```



```

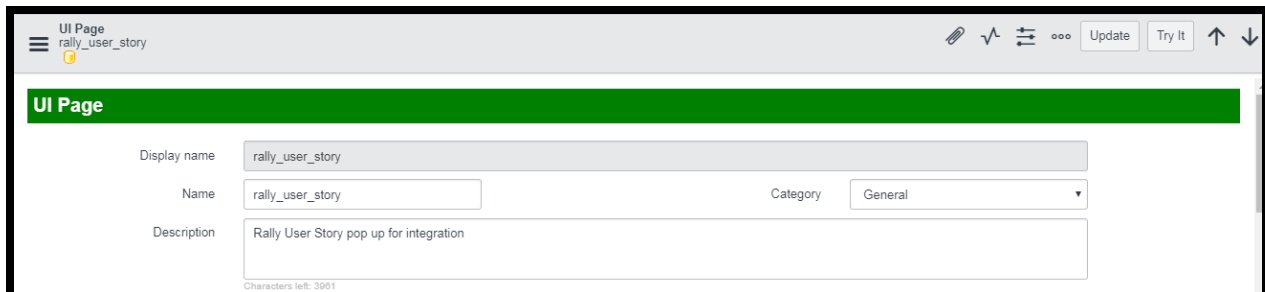
// Pass all the values
dialog.setSize(1500,1500);
dialog.setTitle('Create User Story');
dialog.setPreference("sysparm_name", name);
dialog.setPreference("sysparm_stakeholder",stakeholder);
//dialog.setPreference("sysparm_notes_text", notes);
dialog.setPreference("sysparm_table_id",tableID);
dialog.setPreference("sysparm_description", description);
dialog.render();
return false;

//g_form.addInfoMessage("+g_form.getDisplayBox('variables.sbr_requester_id').value);
//g_form.addInfoMessage("+g_form.getValue('variables.short_description'));
//g_form.addInfoMessage("+g_form.getValue('variables.description'));

/*
var fetchVariables = new GlideAjax('AjaxCheckCustomers2');
fetchVariables.addParam('sysparm_name','getVariables');
fetchVariables.addParam('sysparm_request', g_form.getValue('request'));
fetchVariables.getXML(sendMessage);
*/
}

```

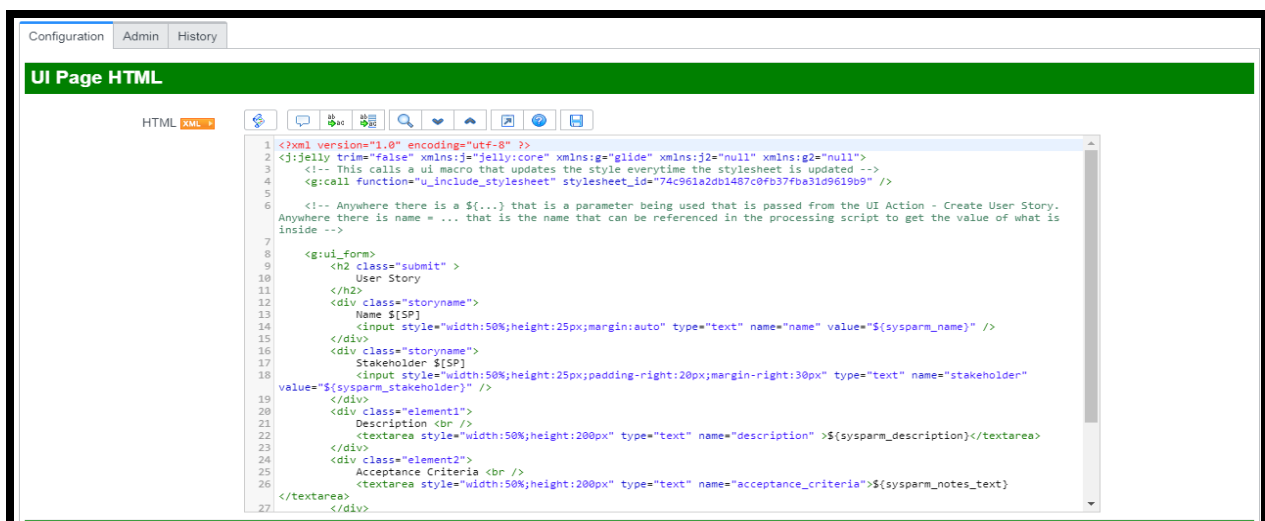
UI Page – rally_user_story: This contains all of the script in the processing script section that actually makes the calls to Rally through REST for creating a user story



The screenshot shows the configuration page for a UI Page named 'rally_user_story'. The page has a green header with the title 'UI Page'. Below the header, there are several input fields and a dropdown menu:

- Display name:** rally_user_story
- Name:** rally_user_story
- Category:** General (dropdown menu)
- Description:** Rally User Story pop up for integration

At the bottom left, it says 'Characters left: 2001'.



The screenshot shows the HTML configuration for the UI Page. The page has a green header with the title 'UI Page HTML'. Below the header, there are tabs for 'Configuration', 'Admin', and 'History'. The main content area shows the HTML code for the UI Page, which includes a form for creating a user story. The code is as follows:

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <j:jquery trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">
3   <!-- This calls a ui macro that updates the style everytime the stylesheet is updated -->
4   <g:call function="u_include_stylesheet" stylesheet_id="74c961a2db1487c0fb37fba31d9619b9" />
5
6   <!-- Anywhere there is a ${...} that is a parameter being used that is passed from the UI Action - Create User Story.
7   Anywhere there is name = ... that is the name that can be referenced in the processing script to get the value of what is
8   inside -->
9
10  <g:ui_form>
11    <h2 class="submit" >
12      User Story
13    </h2>
14    <div class="storyname">
15      Name ${SP}
16    </div>
17    <div class="storyname">
18      Stakeholder ${SP}
19    </div>
20    <div class="element1">
21      Description <br />
22    </div>
23    <div class="element2">
24      Acceptance Criteria <br />
25    </div>
26  </g:ui_form>

```

```

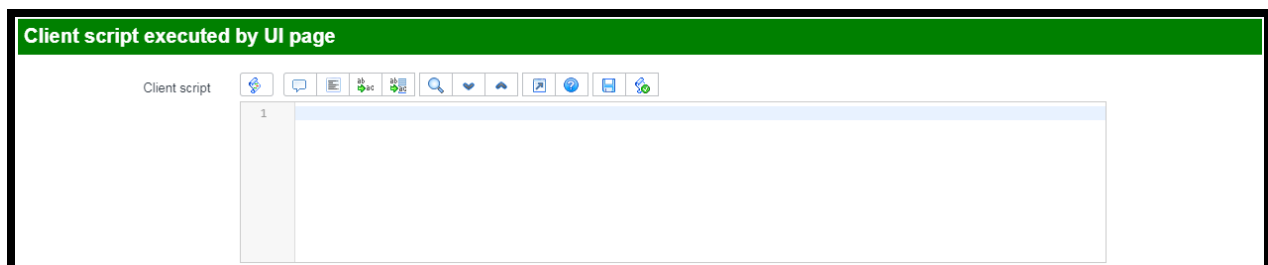
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">
  <!-- This calls a ui macro that updates the style everytime the stylesheet is updated -->
  <g:call function="u_include_stylesheet" stylesheet_id="74c961a2db1487c0fb37fba31d9619b9" />

  <!-- Anywhere there is a ${...} that is a parameter being used that is passed from the UI Action - Create User Story.
  Anywhere there is name = ... that is the name that can be referenced in the processing script to get the
  value of what is inside -->

  <g:ui_form>
    <h2 class="submit" >
      User Story
    </h2>
    <div class="storyname">
      Name ${SP}
      <input style="width:50%;height:25px;margin:auto" type="text" name="name"
value="${sysparm_name}" />
    </div>
    <div class="storyname">
      Stakeholder ${SP}
      <input style="width:50%;height:25px;padding-right:20px;margin-right:30px" type="text"
name="stakeholder" value="${sysparm_stakeholder}" />
    </div>
    <div class="element1">
      Description <br />
      <textarea style="width:50%;height:200px" type="text" name="description"
>${sysparm_description}</textarea>
    </div>
    <div class="element2">
      Acceptance Criteria <br />
      <textarea style="width:50%;height:200px" type="text"
name="acceptance_criteria">${sysparm_notes_text}</textarea>
    </div>
    <div class="submit">
      <g:dialog_buttons_ok_cancel ok="return true" />
    </div>
    <!-- Secretely pass the sys_id of the RITM -->
    <input type="hidden" name="tableID" value="${sysparm_table_id}" />

  </g:ui_form>
</j:jelly>

```



Processing script executed by UI page

Processing script

```

1 // Has nothing to do with the integration, returns user to the RITM after submitting
2 var urlOnStack = GlideSession.get().getStack().bottom();
3 response.sendRedirect(urlOnStack);
4
5 // Everything for integration happens in this try
6 try {
7     // Create the REST message to create user story
8     var story = new sn_ws.RESTMessageV2();
9     story.setHttpMethod('post');
10    // Pass the API key in header to clear authentication
11    story.setRequestHeader('ZSESSIONID', ' ');
12    story.setEndpoint('https://rally1.rallydev.com/slm/webservice/v2.0/hierarchicalrequirement/create');
13
14    // This is the content of the message
15    var storyBody = '{"HierarchicalRequirement": {"Name":"${name}", "Project":"140242043552",
16    "Description":"${description}", "c_SNOWID":"${stakeholder}", "c_SNOWLinkText":"${snow_link}",
17    "c_SNOWIntegrationID":"${sys_id}"}';
18
19    story.setRequestBody(storyBody);
20    // Name of story
21    story.setStringParameter('name', name);
22    // Content of Description
23    story.setStringParameter('description', description);
24    // Name of Stakeholder
25    story.setStringParameter('stakeholder', stakeholder);
26    // SNOW sys_id
27    story.setStringParameter('sys_id', tableID);
28    // SNOW link
29    story.setStringParameter('snow_link',gs.getProperty('glide.servlet.uri')+urlOnStack);
30
31    // Send message and get status and response

```

Update Try It

Related Links
[Personalize all](#)
[Add to Update Set](#)

```

// Has nothing to do with the integration, returns user to the RITM after submitting
var urlOnStack = GlideSession.get().getStack().bottom();
response.sendRedirect(urlOnStack);

// Everything for integration happens in this try
try {
    // Create the REST message to create user story
    var story = new sn_ws.RESTMessageV2();
    story.setHttpMethod('post');
    // Pass the API key in header to clear authentication
    story.setRequestHeader('ZSESSIONID', ' ');
    story.setEndpoint('https://rally1.rallydev.com/slm/webservice/v2.0/hierarchicalrequirement/create');

    // This is the content of the message
    var storyBody = '{"HierarchicalRequirement": {"Name":"${name}", "Project":"140242043552",
    "Description":"${description}", "c_SNOWID":"${stakeholder}", "c_SNOWLinkText":"${snow_link}",
    "c_SNOWIntegrationID":"${sys_id}"}';

    story.setRequestBody(storyBody);
    // Name of story
    story.setStringParameter('name', name);
    // Content of Description
    story.setStringParameter('description', description);
    // Name of Stakeholder
    story.setStringParameter('stakeholder', stakeholder);
    // SNOW sys_id
    story.setStringParameter('sys_id', tableID);
    // SNOW link
    story.setStringParameter('snow_link',gs.getProperty('glide.servlet.uri')+urlOnStack);

    // Send message and get status and response
    var response = story.execute();

```

```
var responseBody = response.getBody();
var httpStatus = response.getStatusCode();
gs.log("httpStatus: " + httpStatus.toString());
gs.log("Response: " + responseBody);

// This finds the reference url of the newly created object
var startIndex = responseBody.search("_ref:");
var endIndex = responseBody.search("_refObjectUUID");
var artifactID = responseBody.substring(startIndex+9,endIndex);
gs.log(artifactID);

// This determines if there are any attachments for this RITM
var target = new GlideRecord('sys_attachment');
target.addQuery('table_sys_id', tableID);
target.query();
var base64Data;
// This encodes the attachment into base64 binary so that it can be turned into attachment content by Rally
while(target.next()) {
    var fileName = target.file_name;
    var sa = new GlideSysAttachment();
    var binData = sa.getBytes(target);
    base64Data = GlideStringUtil.base64Encode(binData);
    // This is the REST message that creates the actual attachment content
    var attachmentContentMessage = new sn_ws.RESTMessageV2();
    attachmentContentMessage.setRequestHeader('ZSESSIONID', '_blahblahblah');

    attachmentContentMessage.setEndpoint('https://rally1.rallydev.com/slm/webservice/v2.0/attachmentcontent/create');

    attachmentContentMessage.setHttpMethod('put');

    var attachContent = '{"AttachmentContent":{"Content":"${content}"}';
    attachmentContentMessage.setRequestBody(attachContent);
    attachmentContentMessage.setStringParameter('content', base64Data);

    var response = attachmentContentMessage.execute();
    responseBody = response.getBody();
    httpStatus = response.getStatusCode();
    gs.log("httpStatusss: " + httpStatus.toString());
    gs.log("ResponseBody: " + responseBody);

    // Get the reference of the attachmentContent
    startIndex = responseBody.search("_ref:");
    endIndex = responseBody.search("_refObjectUUID");
    var contentID = responseBody.substring(startIndex+9,endIndex);
    gs.log(contentID);

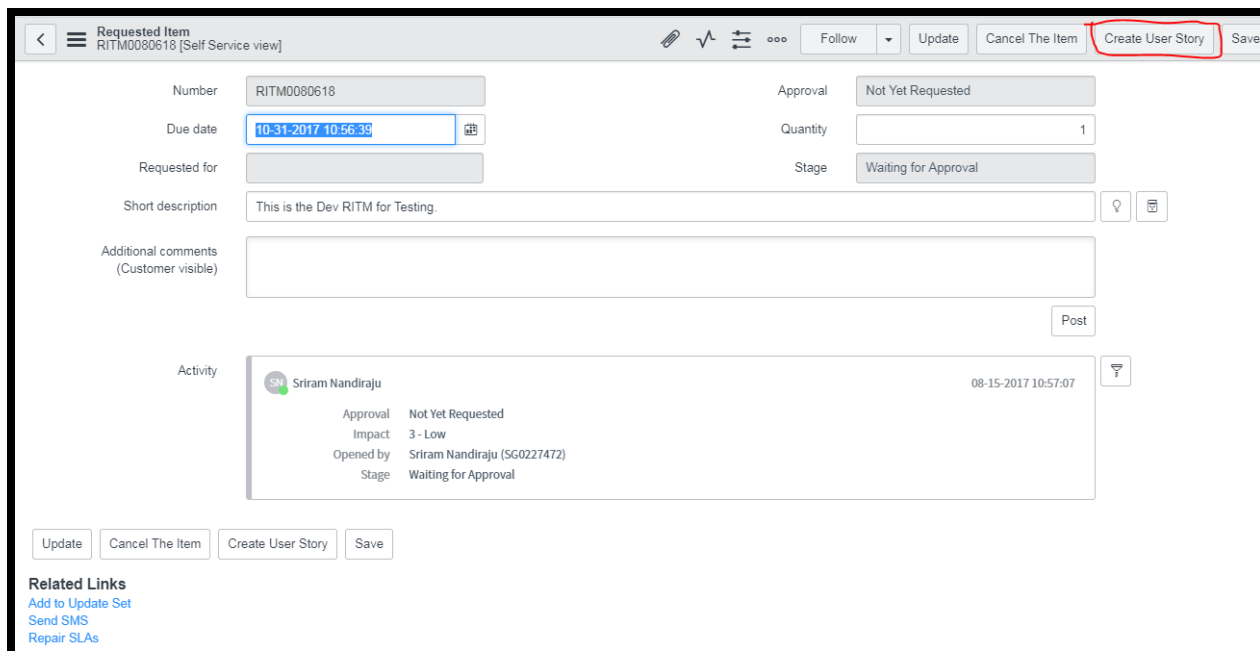
    // This message sends the actual attachment to User Story
    var attachment = new sn_ws.RESTMessageV2();
    attachment.setRequestHeader('ZSESSIONID', '_blahblahblah');
    attachment.setEndpoint('https://rally1.rallydev.com/slm/webservice/v2.0/attachment/create');
    attachment.setHttpMethod('post');

    var body = '{"Attachment":{"Content":"${contentID}", "Artifact":"${artifactID}", "Name": "${file_name}",
"ContentType": "application/octet-stream"}';
    attachment.setRequestBody(body);
    attachment.setStringParameter('contentID', contentID);
    attachment.setStringParameter('artifactID', artifactID);
```

```
attachment.setStringParameter('file_name', fileName);

var response = attachment.execute();
responseBody = response.getBody();
httpStatus = response.getStatusCode();
gs.log("httpStatus: " + httpStatus.toString());
gs.log("Response: " + responseBody);
startIndex = responseBody.search("_refObjectUUID:");
endIndex = responseBody.search(", "_objectVersion");
var rallySysID = responseBody.substring(startIndex+19,endIndex);
gs.log(rallySysID);
// set the rally sys id of the attachment in servicenow to its objectID
target.u_rally_sys_id=rallySysID;
target.insert();
}
}
catch(ex) {
    var message = ex.getMessage();
}
}
```

Create New “Requested Item”



Requested Item
RITM0080618 [Self Service view]

Number: RITM0080618 Approval: Not Yet Requested

Due date: 10-31-2017 10:56:39 Quantity: 1

Requested for: Stage: Waiting for Approval

Short description: This is the Dev RITM for Testing.

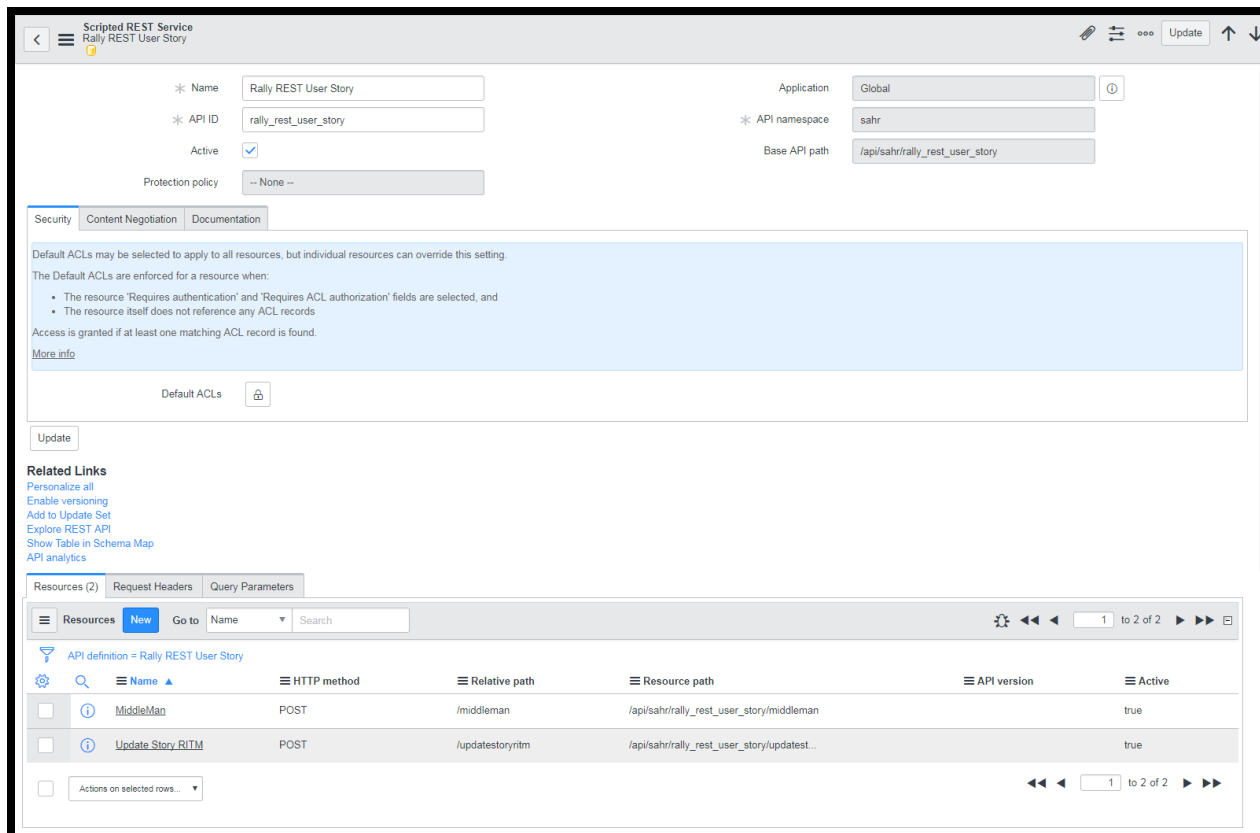
Additional comments (Customer visible):

Activity:

- Sriram Nandiraju (08-15-2017 10:57:07)
 - Approval: Not Yet Requested
 - Impact: 3 - Low
 - Opened by: Sriram Nandiraju (SG0227472)
 - Stage: Waiting for Approval

Buttons: Update, Cancel The Item, **Create User Story**, Save

Related Links
[Add to Update Set](#)
[Send SMS](#)
[Repair SLAs](#)



Scripted REST Service
Rally REST User Story

Name: Rally REST User Story Application: Global

API ID: rally_rest_user_story API namespace: sahr

Active: Base API path: /api/sahr/rally_rest_user_story

Protection policy: -- None --

Security | Content Negotiation | Documentation

Default ACLs may be selected to apply to all resources, but individual resources can override this setting. The Default ACLs are enforced for a resource when:

- The resource 'Requires authentication' and 'Requires ACL authorization' fields are selected, and
- The resource itself does not reference any ACL records

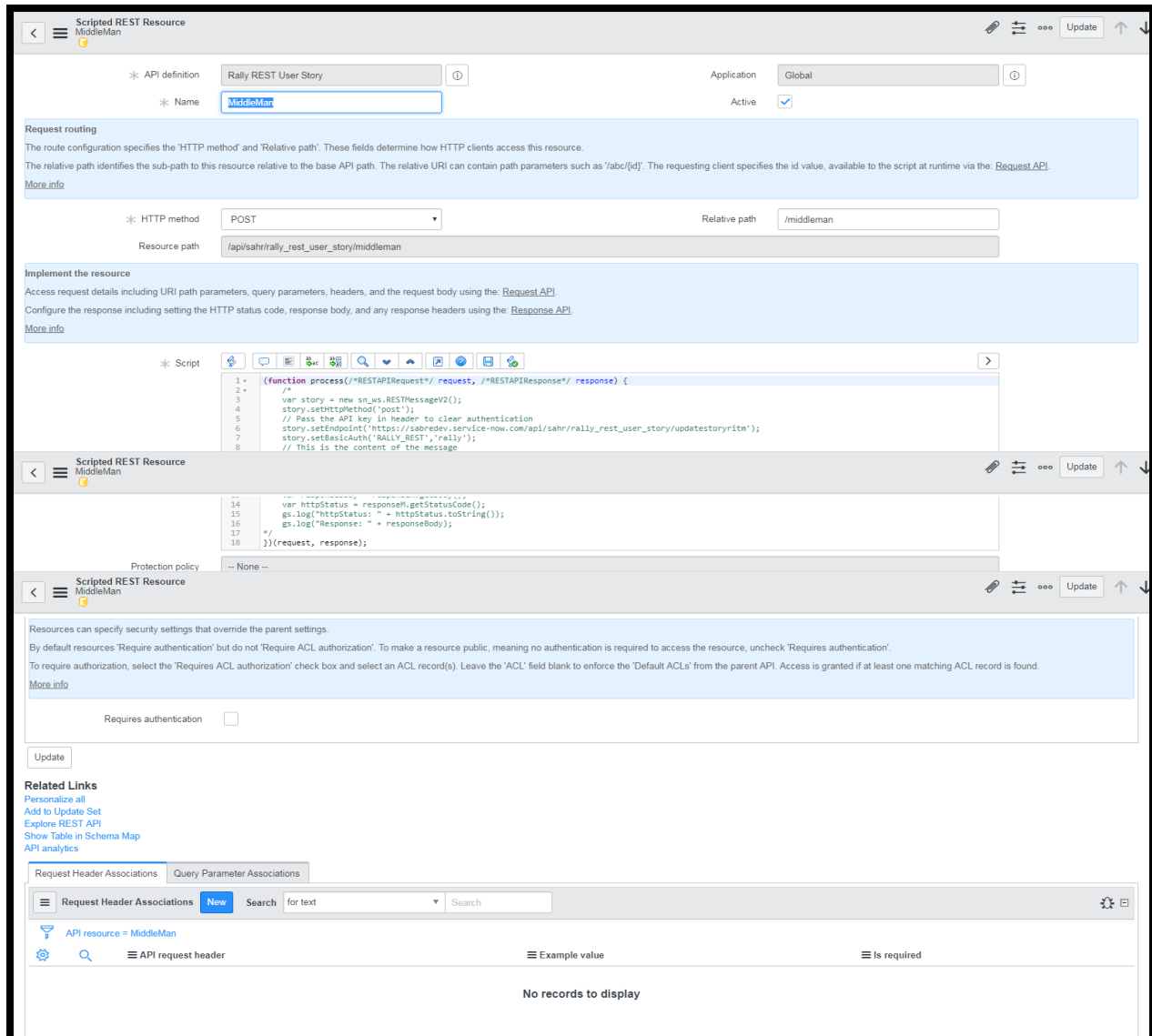
Access is granted if at least one matching ACL record is found.

Default ACLs

Related Links
[Personalize all](#)
[Enable versioning](#)
[Add to Update Set](#)
[Explore REST API](#)
[Show Table in Schema Map](#)
[API analytics](#)

Resources (2) | Request Headers | Query Parameters

Name	HTTP method	Relative path	Resource path	API version	Active
MiddleMan	POST	/middleman	/api/sahr/rally_rest_user_story/middleman		true
Update Story.RITM	POST	/updatestoryritm	/api/sahr/rally_rest_user_story/updatest...		true



Scripted REST Resource
MiddleMan

* API definition: Rally REST User Story Application: Global

* Name: MiddleMan Active:

Request routing
The route configuration specifies the 'HTTP method' and 'Relative path'. These fields determine how HTTP clients access this resource.
The relative path identifies the sub-path to this resource relative to the base API path. The relative URI can contain path parameters such as '/abc/{id}'. The requesting client specifies the id value, available to the script at runtime via the: [Request API](#).
[More info](#)

* HTTP method: POST Relative path: /middleman

Resource path: /api/sahr/rally_rest_user_story/middleman

Implement the resource
Access request details including URI path parameters, query parameters, headers, and the request body using the: [Request API](#).
Configure the response including setting the HTTP status code, response body, and any response headers using the: [Response API](#).
[More info](#)

* Script

```

1 * (function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
2 *   /*
3 *   var story = new sn_ws.RESTMessageV2();
4 *   story.setHttpMethod('post');
5 *   // Pass the API key in header to clear authentication
6 *   story.setEndpoint('https://sabredev.service-now.com/api/sahr/rally_rest_user_story/updatestoryritm');
7 *   story.setBasicAuth('RALLY_REST', 'rally');
8 *   // This is the content of the message
9 *   var requestBody = request.body.dataString;
10 *
11 *   // Send message and get status and response
12 *   var responseM = story.execute();
13 *   var responseBody = responseM.getBody();
14 *   var httpStatus = responseM.getStatusCode();
15 *   gs.log("httpStatus: " + httpStatus.toString());
16 *   gs.log("Response: " + responseBody);
17 *
18 * })(request, response);

```

Protection policy: -- None --

Resources can specify security settings that override the parent settings.
By default resources 'Require authentication' but do not 'Require ACL authorization'. To make a resource public, meaning no authentication is required to access the resource, uncheck 'Requires authentication'.
To require authorization, select the 'Requires ACL authorization' check box and select an ACL record(s). Leave the 'ACL' field blank to enforce the 'Default ACLs' from the parent API. Access is granted if at least one matching ACL record is found.
[More info](#)

Requires authentication:

Update

Related Links
[Personalize all](#)
[Add to Update Set](#)
[Explore REST API](#)
[Show Table in Schema Map](#)
[API analytics](#)

Request Header Associations Query Parameter Associations

Request Header Associations New Search: for text Search

API resource = MiddleMan

API request header Example value Is required

No records to display

```

(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
  /*
  var story = new sn_ws.RESTMessageV2();
  story.setHttpMethod('post');
  // Pass the API key in header to clear authentication
  story.setEndpoint('https://sabredev.service-now.com/api/sahr/rally_rest_user_story/updatestoryritm');
  story.setBasicAuth('RALLY_REST', 'rally');
  // This is the content of the message
  var requestBody = request.body.dataString;

  // Send message and get status and response
  var responseM = story.execute();
  var responseBody = responseM.getBody();
  var httpStatus = responseM.getStatusCode();
  gs.log("httpStatus: " + httpStatus.toString());
  gs.log("Response: " + responseBody);

  */
})(request, response);

```

Scripted REST Resource
Update Story RITM

* API definition: Rally REST User Story Application: Global

* Name: Update Story RITM Active:

Request routing

The route configuration specifies the 'HTTP method' and 'Relative path'. These fields determine how HTTP clients access this resource.

The relative path identifies the sub-path to this resource relative to the base API path. The relative URI can contain path parameters such as '/abc/{id}'. The requesting client specifies the id value, available to the script at runtime via the [Request API](#).

[More info](#)

* HTTP method: POST Relative path: /updatestoryritm

Resource path: /api/sahr/rally_rest_user_story/updatestoryritm

Implement the resource

Access request details including URI path parameters, query parameters, headers, and the request body using the [Request API](#).

Configure the response including setting the HTTP status code, response body, and any response headers using the [Response API](#).

[More info](#)

* Script

```

1 (function process(/**RESTAPIRequest*/ request, /**RESTAPIResponse*/ response) {
2   var storeBody = request.body.dataString;
3
4   // These are the fields in snow that are received from rally
5   var fieldsFromRally = [{"sys_id":"e172ce22-9ec8-4b65-9595-08eb1665b90c", /*notes */"abb5af67-e70a-4a98-ad21-26f71b4fe83",
6   /*attachments */ "8b682afb-fc71-4b37-b840-144270cc6ff3", /*workspace*/ "17f72ea6-04fd-449e-925c-89ac079c0668"}];
7   gs.log(storeBody);
8   var parser = JSON.parse(storeBody);

```

```

14   var changes = parser.message.changes;
15
16   var added = changes[fieldsFromRally[2]].added;
17   var removed = changes[fieldsFromRally[2]].removed;
18   if(added){
19     //gs.log('added[0].ref: ' + added[0].ref);
20     //gs.log('added[0]: ' + added[0]);
21     //gs.log('added.ref: ' + added.ref);
22     var addAttach = new sn_us.RESTMessageV2();
23     addAttach.setHttpMethod('get');
24     // Pass the API key in header to clear authentication
25     addAttach.setRequestHeader('ZSESSIONID', 'XXXXXXXXXXXXXXXXXXXX');
26     addAttach.setEndpoint(added[0].ref);
27
28     var addAttachSecond = addAttach.execute();
29     var responseAttachSecond = addAttachSecond.getBody();
30     var httpStatusAttachSecond = addAttachSecond.getStatusCode();
31

```

Protection policy: -- None --

Scripted REST Resource
Update Story RITM

Resources can specify security settings that override the parent settings.

By default resources 'Require authentication' but do not 'Require ACL authorization'. To make a resource public, meaning no authentication is required to access the resource, uncheck 'Requires authentication'.

To require authorization, select the 'Requires ACL authorization' check box and select an ACL record(s). Leave the 'ACL' field blank to enforce the 'Default ACLs' from the parent API. Access is granted if at least one matching ACL record is found.

[More info](#)

Requires authentication:

Related Links

- [Personalize all](#)
- [Add to Update Set](#)
- [Explore REST API](#)
- [Show Table in Schema Map](#)
- [API analytics](#)

Request Header Associations Query Parameter Associations

Search: for text Search

API resource = Update Story RITM

API request header	Example value	Is required
No records to display		

The Rally Rest POST request payload contains the UUID for c_SNOWIntegrationID

```

"17f72ea6-04fd-449e-925c-89ac079c0668":{
  "value":{
    "name":"Integration Workspace",

```



```

"ref": "https://rally1.rallydev.com/slm/webservice/v2.x/workspace/ee5948bb-365c-4869-bcbf-4c5237e4a3e4",
"detail_link": "https://rally1.rallydev.com/slm/#/detail/workspace/30637201215",
"id": "ee5948bb-365c-4869-bcbf-4c5237e4a3e4",
"object_type": "Workspace"
},
"type": "Workspace",
"name": "Workspace",
"display_name": "Workspace",
"ref": null
},
"e172ce22-9ec8-4b65-9595-08eb1665b90c": {
"value": "57752e07db64070075dd9837db9619c8",
"type": "String",
"name": "c_SNOWIntegrationID",
"display_name": "SNOW Integration ID",
"ref": null
},
}

```

<https://rally1.rallydev.com/slm/webservice/v2.0/TypeDefinition/30637201358/Attributes?start=40>

```

{
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/attributedefinition/-10502",
  "_refObjectUUID": "abb5af67-e70a-4a98-ad21-26f71bf4fe83",
  "_objectVersion": "1",
  "_refObjectName": "Notes",
  "CreationDate": "2006-02-11T12:29:05.000Z",
  "_CreatedAt": "Feb 11, 2006",
  "ObjectID": -10502,
  "ObjectUUID": "abb5af67-e70a-4a98-ad21-26f71bf4fe83",
  "VersionId": "1",
  "Subscription": null,
  "Workspace": null,
  "AllowedQueryOperators": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/AttributeDefinition/-10502/AllowedQueryOperators",
    "_type": "AllowedQueryOperator",
    "Count": 2
  },
  "AllowedValueType": null,
  "AllowedValues": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/AttributeDefinition/-10502/AllowedValues",
    "_type": "AllowedAttributeValue",
    "Count": 0
  },
  "AttributeType": "TEXT",
  "Constrained": false,
  "Custom": false,
  "DetailedType": null,
  "ElementName": "Notes",
  "Filterable": true,

```

```

"Hidden": false,
"Hideable": true,
"MaxFractionalDigits": -1,
"MaxLength": 32768,
"Name": "Notes",
"Note": "Artifact notes, which is a rich-text field.",
"Owned": true,
"ReadOnly": false,
"RealAttributeType": "TEXT",
"Required": false,
"Sortable": true,
"SystemRequired": false,
"Type": "string",
"TypeDefinition": {
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/-51004",
  "_refObjectUUID": "d52bf519-e9b8-440d-b653-a8a41f9b93b8",
  "_refObjectName": "Artifact",
  "_type": "TypeDefinition"
},
"VisibleOnlyToAdmins": false,
"_type": "AttributeDefinition"
},

```

<https://rally1.rallydev.com/slm/webservice/v2.0/TypeDefinition/30637201358/Attributes?start=20>

```

{
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/attributedefinition/-17500",
  "_refObjectUUID": "17f72ea6-04fd-449e-925c-89ac079c0668",
  "_objectVersion": "1",
  "_refObjectName": "Workspace",
  "CreationDate": "2006-02-11T12:29:05.000Z",
  "_CreatedAt": "Feb 11, 2006",
  "ObjectID": -17500,
  "ObjectUUID": "17f72ea6-04fd-449e-925c-89ac079c0668",
  "VersionId": "1",
  "Subscription": null,
  "Workspace": null,
  "AllowedQueryOperators": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/AttributeDefinition/-17500/AllowedQueryOperators",
    "_type": "AllowedQueryOperator",
    "Count": 0
  },
  "AllowedValueType": {
    "_rallyAPIMajor": "2",
    "_rallyAPIMinor": "0",
    "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/27154375654",
    "_refObjectUUID": "686004a9-5ca1-43bb-aa85-687ba59f6179",
    "_refObjectName": "Workspace",
    "_type": "TypeDefinition"
  }
}

```

```

},
"AllowedValues": {
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/AttributeDefinition/-17500/AllowedValues",
  "_type": "AllowedAttributeValue",
  "Count": 0
},
"AttributeType": "OBJECT",
"Constrained": false,
"Custom": false,
"DetailedType": null,
"ElementName": "Workspace",
"Filterable": false,
"Hidden": false,
"Hideable": true,
"MaxFractionalDigits": -1,
"MaxLength": -1,
"Name": "Workspace",
"Note": "A reference to the workspace an object is associated with. This field can only be set during object creation.",
"Owned": false,
"ReadOnly": false,
"RealAttributeType": "OBJECT",
"Required": false,
"Sortable": false,
"SystemRequired": false,
"Type": "Workspace",
"TypeDefinition": {
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/-51003",
  "_refObjectUUID": "f6e082b8-df17-43a5-b972-1df1549f217f",
  "_refObjectName": "Workspace Domain Object",
  "_type": "TypeDefinition"
},
"VisibleOnlyToAdmins": false,
"_type": "AttributeDefinition"
},

```

<https://rally1.rallydev.com/slm/webservice/v2.0/TypeDefinition/30637201358/Attributes?start=30>

```

{
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/attributedefinition/-11006",
  "_refObjectUUID": "8b682afb-fc71-4b37-b840-144270cc6ff3",
  "_objectVersion": "1",
  "_refObjectName": "Attachments",
  "CreationDate": "2006-02-11T12:29:05.000Z",
  "_CreatedAt": "Feb 11, 2006",
  "ObjectID": -11006,
  "ObjectUUID": "8b682afb-fc71-4b37-b840-144270cc6ff3",
  "VersionId": "1",
  "Subscription": null,
  "Workspace": null,
  "AllowedQueryOperators": {

```

```
"_rallyAPIMajor": "2",
"_rallyAPIMinor": "0",
"_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/AttributeDefinition/-11006/AllowedQueryOperators",
"_type": "AllowedQueryOperator",
"Count": 4
},
"AllowedValueType": {
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/27154375539",
  "_refObjectUUID": "c85b1559-ef7f-4bd8-acb5-fd1f33e472a1",
  "_refObjectName": "Attachment",
  "_type": "TypeDefinition"
},
"AllowedValues": {
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/AttributeDefinition/-11006/AllowedValues",
  "_type": "AllowedAttributeValue",
  "Count": 0
},
"AttributeType": "COLLECTION",
"Constrained": false,
"Custom": false,
"DetailedType": null,
"ElementName": "Attachments",
"Filterable": true,
"Hidden": false,
"Hideable": true,
"MaxFractionalDigits": -1,
"MaxLength": -1,
"Name": "Attachments",
"Note": "Attachments associated with this requirement",
"Owned": false,
"ReadOnly": false,
"RealAttributeType": "COLLECTION",
"Required": false,
"Sortable": false,
"SystemRequired": false,
"Type": "RequirementAttachmentsType",
"TypeDefinition": {
  "_rallyAPIMajor": "2",
  "_rallyAPIMinor": "0",
  "_ref": "https://rally1.rallydev.com/slm/webservice/v2.0/typedefinition/-51005",
  "_refObjectUUID": "f17aa5b1-9974-47a6-9d5c-71a20694608d",
  "_refObjectName": "Requirement",
  "_type": "TypeDefinition"
},
"VisibleOnlyToAdmins": false,
"_type": "AttributeDefinition"
},
```

Scripted Rest API code logic:

```

(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {
    var storeBody = request.body.dataString;

    // These are the fields in snow that are received from rally
    var fieldsFromRally = ["*sys_id*/"e172ce22-9ec8-4b65-9595-08eb1665b90c", /*notes */"abb5af67-e70a-4a98-
ad21-26f71bf4fe83",
    /*attachments */ "8b682afb-fc71-4b37-b840-144270cc6ff3", /*workspace*/ "17f72ea6-04fd-449e-925c-
89ac079c0668"];
    gs.log(storeBody);
    var parser = JSON.parse(storeBody);
    gs.log("TESTING WEBHOOKS");

    var snowID = parser.message.state[fieldsFromRally[0]].value;

    if(snowID){

        var changes = parser.message.changes;

        var added = changes[fieldsFromRally[2]].added;
        var removed = changes[fieldsFromRally[2]].removed;
        if(added){
            //gs.log('added[0].ref: ' + added[0].ref);
            //gs.log('added[0]: ' + added[0]);
            //gs.log('Added.ref: ' + added.ref);
            var addAttach = new sn_ws.RESTMessageV2();
            addAttach.setHttpMethod('get');
            // Pass the API key in header to clear authentication
            addAttach.setRequestHeader('ZSESSIONID', '_blahblahblah');
            addAttach.setEndpoint(added[0].ref);

            var addAttachSecond = addAttach.execute();
            var responseAttachSecond = addAttachSecond.getBody();
            var httpStatusAttachSecond = addAttachSecond.getStatusCode();
            //gs.log("httpStatusAttachSecond: " + httpStatusAttachSecond.toString());
            //gs.log("responseAttachSecond: " + responseAttachSecond);

            var getAttachContent = JSON.parse(responseAttachSecond);

            var attachContentURL = getAttachContent.Attachment.Content._ref;
            var contentType = getAttachContent.Attachment.ContentType;
            var fileName = getAttachContent.Attachment.Name;

            //gs.log('Attach Content URL: ' + attachContentURL);

            var addDecode = new sn_ws.RESTMessageV2();
            addDecode.setHttpMethod('get');
            // Pass the API key in header to clear authentication
            addDecode.setRequestHeader('ZSESSIONID', '_blahblahblah');
            addDecode.setEndpoint(attachContentURL);

            var addDecodeValue = addDecode.execute();
            var decodedResponse = addDecodeValue.getBody();
            var decodedStatus = addDecodeValue.getStatusCode();
            //gs.log("decodedStatus: " + decodedStatus.toString());
            //gs.log("decodedResponse: " + decodedResponse);
        }
    }
}

```

```

        var getDecode = JSON.parse(decodedResponse);
        if(!getDecode.AttachmentContent.Errors){
            var decodedContent = getDecode.AttachmentContent.Content;

            var requestBodyString ='<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ecc="http://www.service-
now.com/ecc_queue"><soapenv:Header
/><soapenv:Body><ecc:insert><agent>AttachmentCreator</agent><topic>AttachmentCreator</topic><name>'+fileName +
':' + contentType + '</name><source>sc_req_item:' + snowID + '</source><payload>' + decodedContent +
'</payload></ecc:insert></soapenv:Body></soapenv:Envelope>';
            //gs.log('STR1: ' + str1);

            // Here is where attachment Creator would go
            var s = new sn_ws.SOAPMessageV2(); //create an empty SOAP message
            s.setBasicAuth('userid','password');
            s.setSOAPAction('http://www.service-now.com/ecc_queue/insert'); //set the SOAP
action to perform
            s.setEndpoint('https://sabredev.service-now.com/ecc_queue.do?SOAP'); //set the
web service provider endpoint

            s.setRequestBody(requestBodyString);

            gs.log('RequestBody: ' + s.getRequestBody());

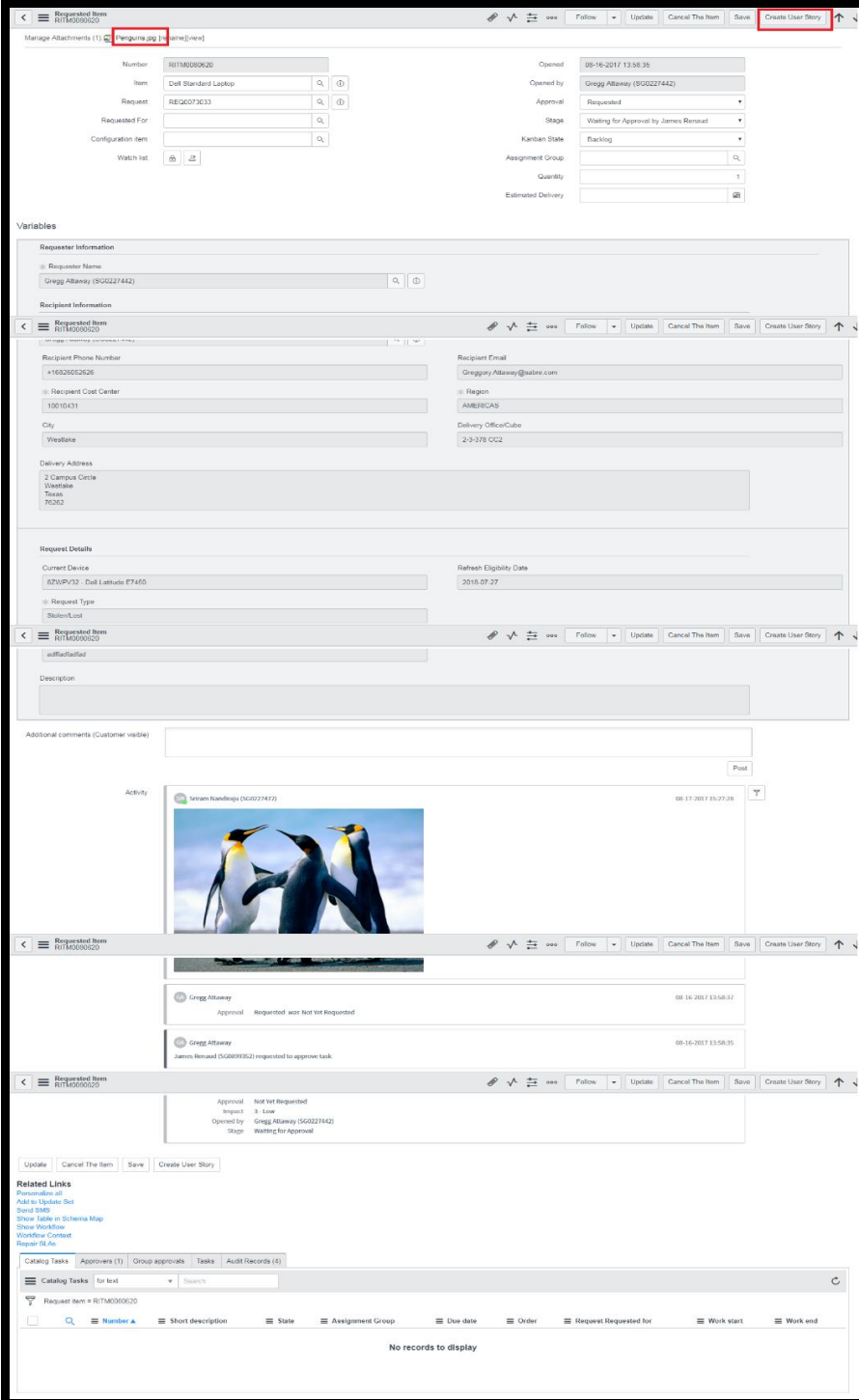
            var attachresponse = s.execute();
            var xmldoc = new XMLDocument(attachresponse.getBody());
            gs.log(xmldoc);
        }
    }
    else if(removed){
        var rallyID = removed.id;
        //gs.log("Made it into removed if");
        var attachment = new GlideRecord('sys_attachment');
        attachment.addQuery('table_sys_id', snowID);
        attachment.addQuery('u_rally_attachment_sys_id', rallyID);
        attachment.query();
        if(attachment.next()){
            attachment.deleteRecord();
        }
    }
    var notes = parser.message.state["abb5af67-e70a-4a98-ad21-26f71bf4fe83"].value;
    gs.log('Notes: ' + notes);
    gs.log("SNOWID Not NULL");
    gs.log('sys_id: ' + snowID);
    var gr = new GlideRecord('sc_req_item');
    gr.addQuery('sys_id', snowID);
    gr.query();
    if(gr.next()){
        gr.comments = notes;
        gr.update();
    }
}

})(request, response);

```

DEMO

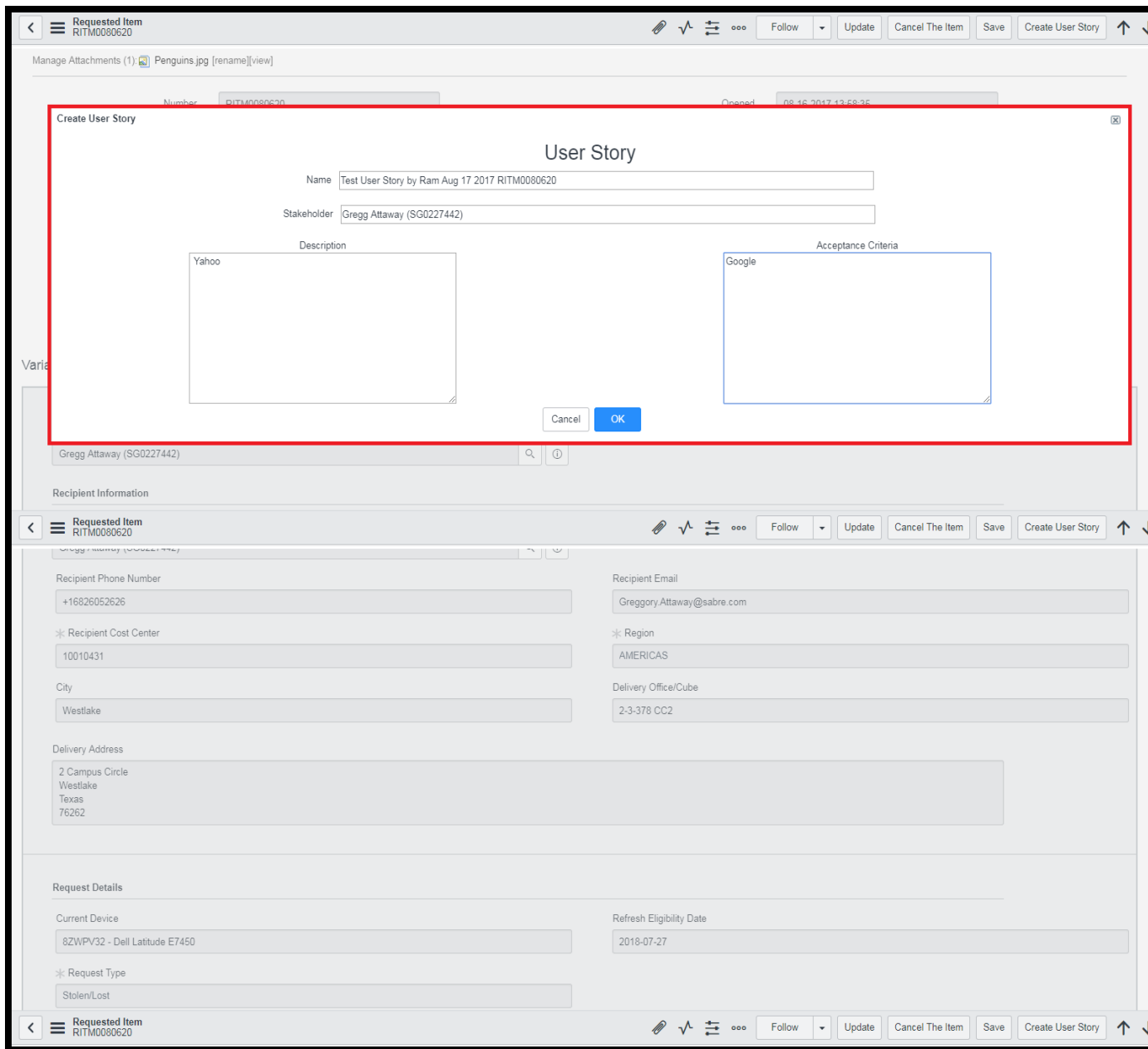
In this demo, we will start with a RITM in Service-Now. Please note the name of the attachment 'Penguins.jpg'. We shall invoke the UI Action by clicking on the button



The screenshot displays the ServiceNow Request Item (RITM) interface for request RITM000020. The top navigation bar shows the attachment 'Penguins.jpg' and the 'Create User Story' button. The main form contains the following sections:

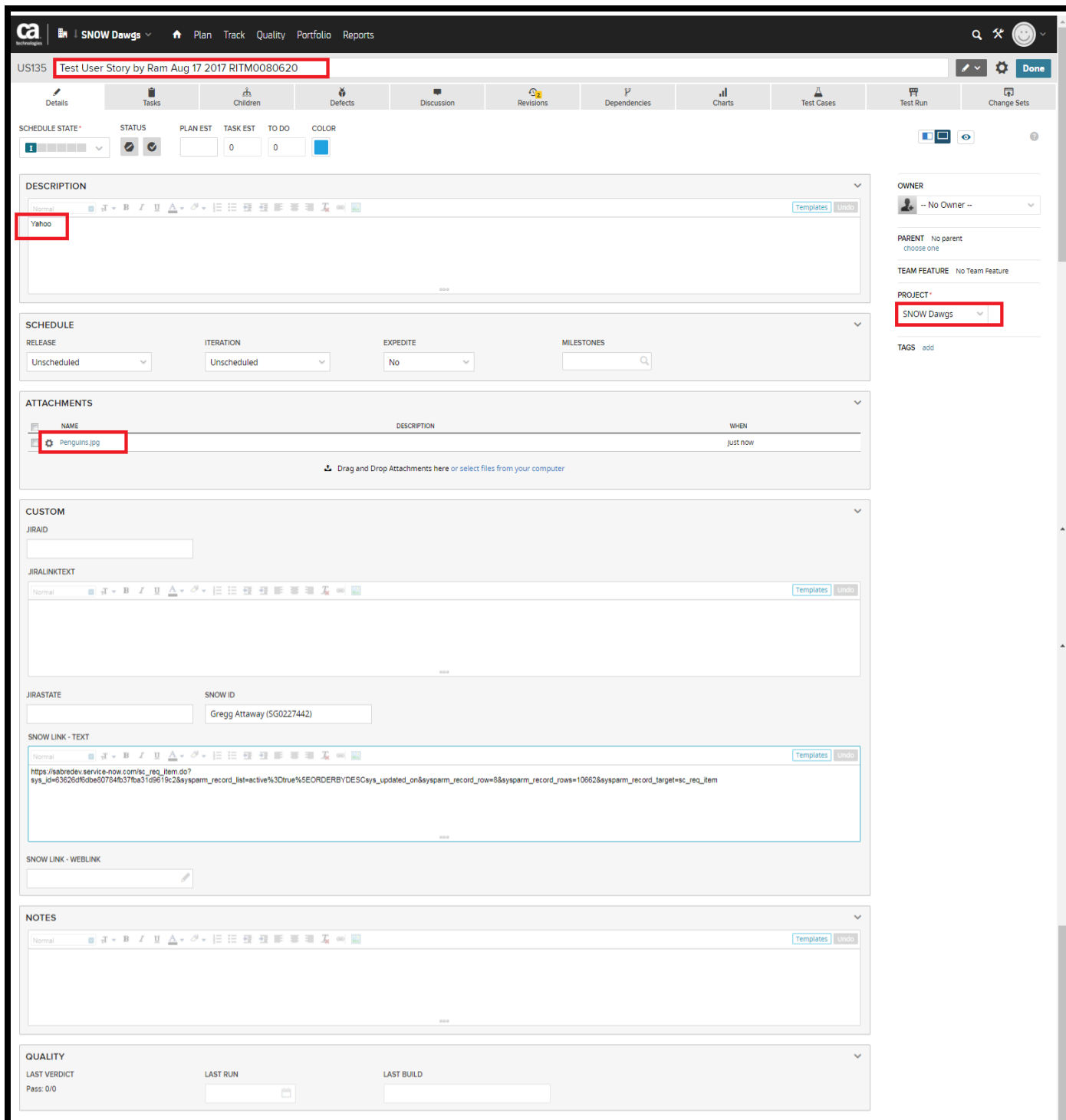
- Request Details:** Number (RITM000020), Item (Dell Standard Laptop), Request (REQ073033), Requested For, Configuration Item, Watch List, Opened (08-16-2017 13:56:35), Opened by (Gregg Altmayer), Requested, Stage (Waiting for Approval by James Renaud), Karbon State (Backlog), Assignment Group, Quantity (1), and Estimated Delivery.
- Variables:**
 - Requester Information:** Requester Name (Gregg Altmayer).
 - Recipient Information:** Recipient Phone Number (+16326052626), Recipient Cost Center (10010431), City (Westlake), Delivery Address (2 Campus Circle, Westlake, Texas 76202), Recipient Email (Greggory.Altmayer@sabre.com), Region (AMERICAS), and Delivery Office/Cube (2-3-378 CC2).
- Request Details:** Current Device (R70VP/32" Dell Latitude E7450), Request Type (Stolen/Lost), and Refresh Eligibility Date (2016-07-27).
- Activity:** A post from Gregg Altmayer (08-17-2017 13:27:36) containing an image of three penguins.
- Approval History:**
 - Gregg Altmayer: Approval Requested, was not yet requested (08-16-2017 13:56:37).
 - Gregg Altmayer: James Renaud (02899362) requested to approve task (08-16-2017 13:56:35).
 - Approval: not yet requested (Impact: Low, Opened by: Gregg Altmayer (500227442), Stage: Waiting for Approval).
- Related Links:** Rescindable, Add to Updater Set, Send SMS, Show Table in Schema Map, Show Workflow, Workflow Contact, Repair SLA.
- Footer:** Catalog Tasks, Approvers (1), Group approvals, Tasks, Audit Records (6), and a table with columns: Number, Short description, State, Assignment Group, Use date, Order, Request Requested for, Work start, Work end. The table currently shows 'No records to display'.

“Create user Story”.

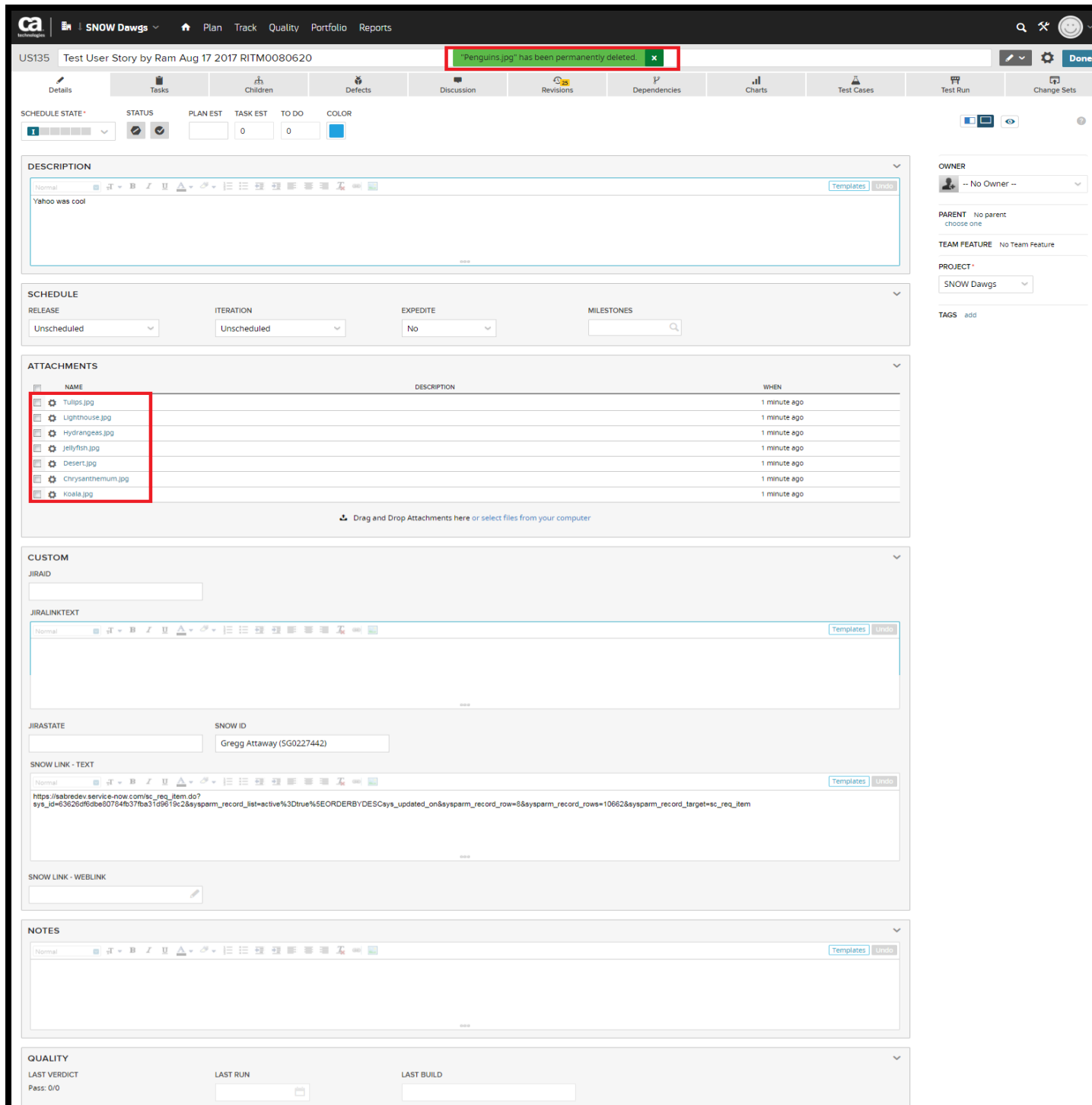


The image shows a screenshot of a web application interface. At the top, there is a header bar with a navigation menu on the left and several action buttons on the right: 'Follow', 'Update', 'Cancel The Item', 'Save', and 'Create User Story'. Below the header, there is a section for 'Manage Attachments (1)' with a thumbnail of 'Penguins.jpg'. The main content area is dominated by a 'Create User Story' dialog box with a red border. This dialog has a title 'User Story' and contains the following fields: 'Name' (Test User Story by Ram Aug 17 2017 RITM0080620), 'Stakeholder' (Gregg Attaway (SG0227442)), 'Description' (Yahoo), and 'Acceptance Criteria' (Google). At the bottom of the dialog are 'Cancel' and 'OK' buttons. Below the dialog, there is a search bar for 'Gregg Attaway (SG0227442)'. The background shows a 'Requested Item' form with sections for 'Recipient Information' (including phone number, email, cost center, region, city, and delivery office), 'Delivery Address' (2 Campus Circle, Westlake, Texas, 76262), and 'Request Details' (Current Device: 8ZWPV32 - Dell Latitude E7450, Refresh Eligibility Date: 2018-07-27, Request Type: Stolen/Lost). The bottom of the page features a navigation bar with the same action buttons as the top.

User Story is successfully created in Rally in “SNOW Dawgs” project. The attachment also got created successfully.



Make some changes in Rally user story.
 Add few attachments
 Delete Penguins.jpg

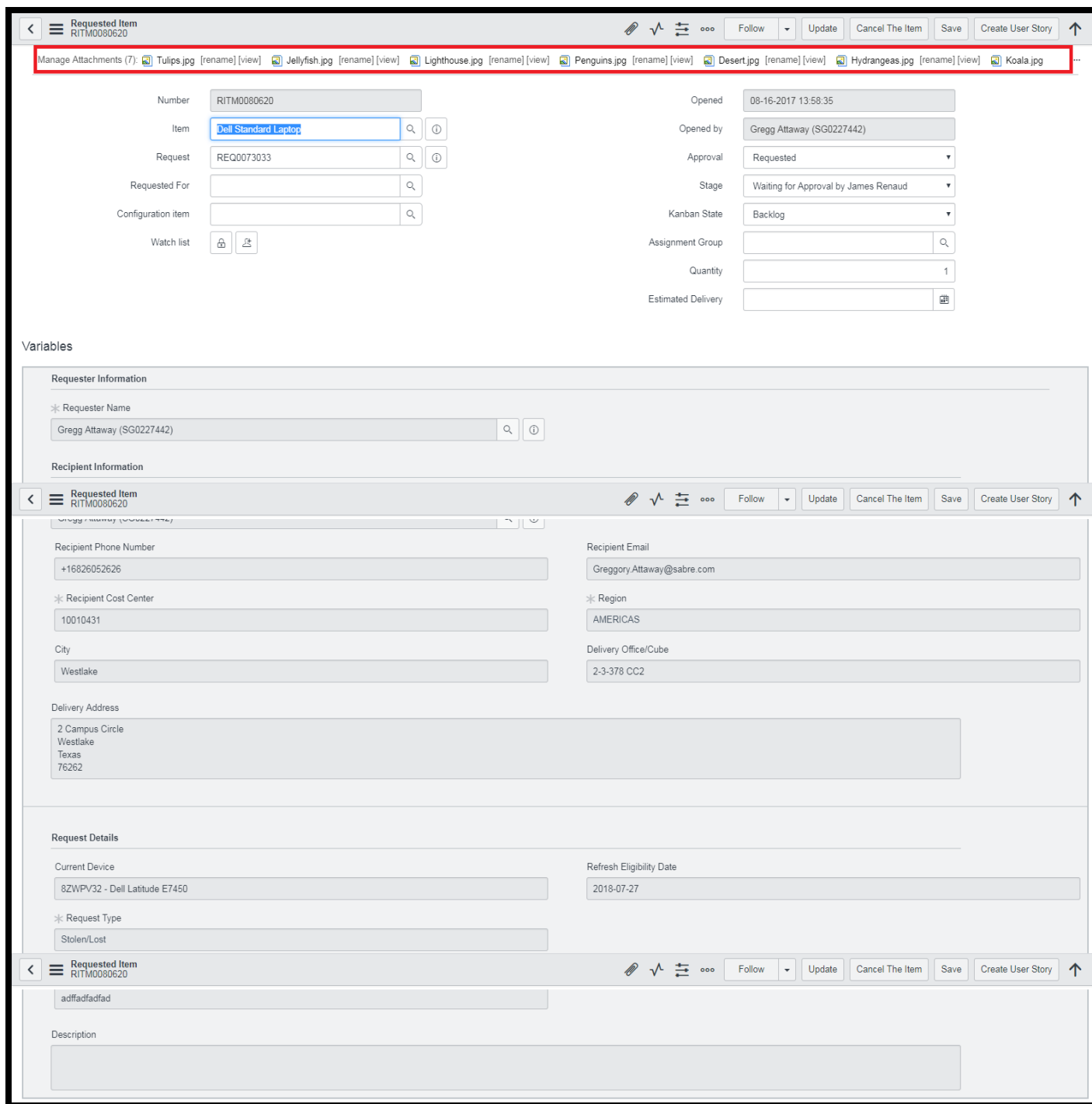


The screenshot shows the Rally user story interface for 'US135 Test User Story by Ram Aug 17 2017 RITM0080620'. A notification at the top states: "Penguins.jpg" has been permanently deleted. The interface includes sections for Description, Schedule, Attachments, Custom, and Notes. The Attachments table lists several files, with 'Penguins.jpg' highlighted in red. The Description field contains the text 'Yahoo was cool'. The Schedule section shows 'Unscheduled' for both Release and Iteration. The Attachments table is as follows:

NAME	DESCRIPTION	WHEN
Tulips.jpg		1 minute ago
Lighthouse.jpg		1 minute ago
Hydrangeas.jpg		1 minute ago
Jellyfish.jpg		1 minute ago
Desert.jpg		1 minute ago
Chrysanthemum.jpg		1 minute ago
Koala.jpg		1 minute ago

The Custom section includes fields for JIRA ID, JIRALINKTEXT, JIRASTATE, SNOW ID (Gregg Attaway (SG0227442)), SNOW LINK - TEXT (a long URL), and SNOW LINK - WEBLINK. The Notes section is empty. The Quality section shows 'LAST VERDICT Pass: 0/0'.

The attachments show up in Service-Now.



The screenshot displays a Service-Now interface for a Requested Item (RITM0080620). At the top, a red box highlights the 'Manage Attachments (7)' section, which lists several image files: Tulips.jpg, Jellyfish.jpg, Lighthouse.jpg, Penguins.jpg, Desert.jpg, Hydrangeas.jpg, and Koala.jpg. Below this, the form is divided into several sections:

- Item Information:** Number (RITM0080620), Item (Dell Standard Laptop), Request (REQ0073033), Requested For, Configuration item, and Watch list.
- Operational Details:** Opened (08-16-2017 13:58:35), Opened by (Gregg Attaway), Approval (Requested), Stage (Waiting for Approval by James Renaud), Kanban State (Backlog), Assignment Group, Quantity (1), and Estimated Delivery.
- Variables Section:**
 - Requester Information:** Requester Name (Gregg Attaway).
 - Recipient Information:** Recipient Phone Number (+16826052626), Recipient Cost Center (10010431), City (Westlake), Recipient Email (Greggory.Attaway@sabre.com), Region (AMERICAS), and Delivery Office/Cube (2-3-378 CC2).
 - Delivery Address:** 2 Campus Circle, Westlake, Texas, 76262.
 - Request Details:** Current Device (8ZWPV32 - Dell Latitude E7450), Refresh Eligibility Date (2018-07-27), and Request Type (Stolen/Lost).
- Description:** A large text area at the bottom containing the placeholder text 'adffadfadfad'.

Known Issues: Below are few of the known issues.

1. When the user makes changes to the form in Rally it does populate in SNOW under comments but it says that the user who made the comments was Guest
2. Duplicate Content (When the same attachment has been used previously) is not accepted by Rally and does not create an attachment on the item.
 - a. Workaround: Put the attachment on the story manually
3. Hard coded values – UUIDs and Projects will need to be consistent and updated as needed.

Scope for Enhancement:

1. Deciding how to handle selecting the project, whether attached to assignment or not.
2. Deciding how to handle attachments, whether to make the changes always go both ways.

Research Items:

1. Do Rally webhooks have any rules on how many times they can be called in a certain time period?
2. Is there any way to customize the payload of a webhook? At this point, the webhook is POSTing all the data items tied to the object in Rally in response to a change in just one single data item within the object in Rally. From a Service-Now integration standpoint, we don't really need to know all the data items. All we need to know is what changed, what the old and new values are!
3. Is there any way to find the UUID of custom made fields? Like a dictionary or query?
4. Whenever we try to create AttachmentContent with content that is the same as previous content it does not let us.
5. Is there a way within rally to monitor webhooks and see when they fire and are called? Like a log system?

THE END