PagerDuty

KTSL Transformation starts here

Pagerduty
Helix Integration Guide

Author: Aaron Perrott

Date: 25/09/2021

Version: 3.0

1 Contents

2 Document Control

2.1 Version Control

Version	Status & summary of changes	Date
Draft	Initial draft	02-Aug-21
1.0	Initial Release	06-Aug-21
2.0	Updated with workflow changes	10-Aug-21
3.0	Enhancements on config, mapping rules	25-Sept-21

2.2 Document Purpose

This document gives detail on the API calls and Helix setup for the integration. It includes the configuration details and the functionality available via the integration.

2.3 Intended Audience

The document is aimed at organisations wishing to use and implement the PagerDuty to Helix integration.

2.4 Confidentiality

This document is confidential.

2.5 Related Documents

The API detail is available in Swagger : Swagger UI (ktsl.com).

Please note the Swagger link is currently on a development server and may not always be available.

3 High Level Overview

3.1 Function

This document outlines the PagerDuty to Helix integration. It includes the functionality available within the interface and the implementation. It will document both the API layer and the Helix layer. The integration has been written to support both On-Premise and Helix implementations of Remedy against the PagerDuty cloud environment.

3.2 Core Components

The integration is provided by two distinct components. A C# .Net core application which acts as a middleware bridge between PagerDuty and Helix and a Helix component used to manage the workflow requirements.

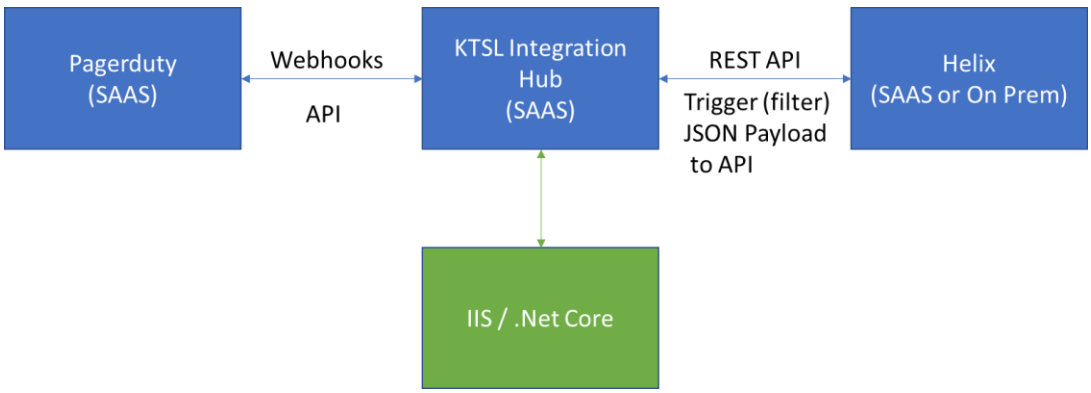
The application brokers the calls between PagerDuty and Helix by wrapping the API's into a methods Helix can call. It also manages Authentication, Config, Retries and Error Handling. By segregating this from Helix it means the integration can leverage the capabilities of an application to manage areas not traditionally a core area of Helix. It also allows logging to be controlled outside of the Helix environment (when Helix is SAAS). It also keeps the footprint on the Helix side as low as possible to ensure support from BMC is not impacted.

This approach also ensures minimal issues during SAAS upgrades from either PagerDuty or BMC. The application is fully supported by PagerDuty in-conjunction with KTSL and will support version upgrades and changes on both sides.

Within the Helix integration all transactions and config are held in new forms. This is to reduce any changes required to the out of the box (OOTB) forms within Helix. It also allows for any unique customer customisations to be left unaffected by creating a layer between the core forms and the PagerDuty integration.

3.2.1 Core Component Diagram

Below shows the core components involved in the integration :



3.3 Functionality

The below tables outlines the high level functionality contained within the integration. Each component is documented in detail within this documentation.

Requirement	Included
Configuration & Service Provisioning (including mappings)	✓
Map Service CI (or other objects) to PD Service and Provisioning Services from Remedy CMDB	✓
Provisioning Users, Teams from Remedy	✓
Trigger Remedy Incidents to Pagerduty (including Description & Priority)	✓
Trigger PagerDuty to Helix (include Description & Priority)	✓
Set Vendor ID in Remedy when initated in Helix	✓
Sync Acknowledgement (PD->R)	✓
Sync Resolve (R->PD & PD->R)	✓
Sync Notes, Status Updates into Work Items	✓
Timeline updates (Responder added, Escalated, Reassigned, Reopened)	✓
Ability to Add Responders, Add Status Updates, Run Response Plays	✓
Priority based routing to different EP's	✓
Priority Updates between PagerDuty and Helix	✓
Reassignment and Delegation rules between PagerDuty and Helix	✓

3.4 Additional Features

In addition to the specific functionality above, the Integration also has the following components :

- Queue Management
- Logging
- Subscription management
- Error handling
- Authentication (via OAUTH or SAML)
- Remedy workflow wrapper

These are all handled within the c# .NET application.

4 Installation and Setup Steps

4.1 Integration Hub

The setup requires a secure network connection over https between the customers Helix instance and the hub. This can be done in a variety of ways and would usually use the same method as accessing the Pagerduty SAAS solution. This can be via a VPN, cloud connectivity or through the Helix client gateway. It will also support any additional middleware component such as Jitterbit, Mulesoft, Software AG etc.

The actual setup of the hub is done as part of the delivery by Pagerduty/KTSL. It will be configured against the customers instance of Pagerduty and the Helix environment. The authentication can be via any OAUTH/SAML mechanism and/or Helix SSO.

4.2 Helix Components

The Helix components are documented within this document. For the installation you will first need to create a set of manual fields on the OOTB forms. The reason for this to be done manually is in case any customisations or changes have been made. The fields have been kept to a minimum in line with the BMC standards.

4.2.1 Manual Created Fields

Within HPD:Help Desk you will need to create a field called z1D Action_PD – 753100001 – Display Only Character field

Within AST:BusinessServices you will need to create a field called z1D Action_PD – 753100001 – Display Only Character field

Within CTM:Support Group you will need to create a field called z1D Action_PD – 753100001 – Display Only Character field

Within HPD:WorkLog you will need to create a field called z1D Action_PD – 753100001 – Display Only Character field

4.2.2 Definition

All other workflow and forms are unique to PagerDuty. You will need to import the provided .def file into Helix through Dev Studio into the Development instance and then via the deployment manager through the environments. The .def file contains all the workflow specific to the integration.

A .arx is also provided with example config data which can be changed to contain the data specific to the customer, but provides a baseline for the required config.

5 Helix Components

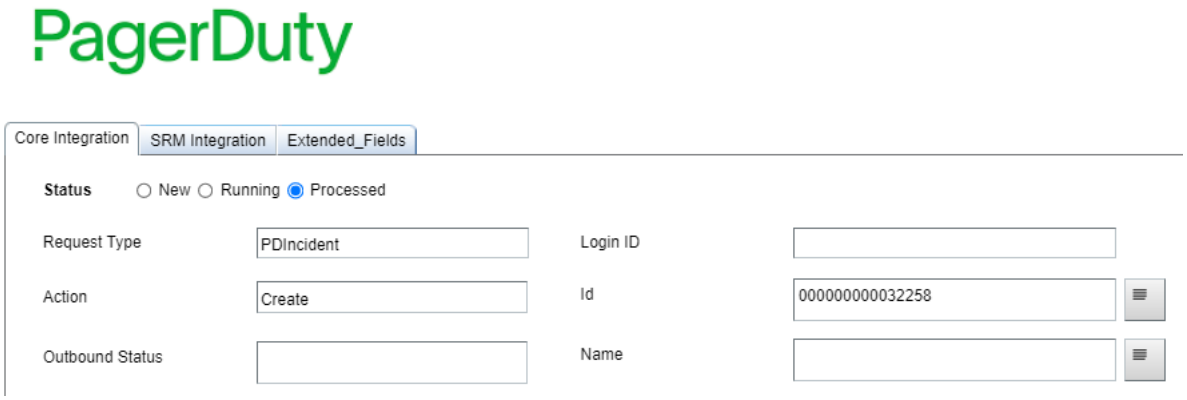
This section will look at the components required from the Helix end of the integration.

5.1 Forms

The definition file that comes with the integration contains the following forms :

- PD:PD:PagerDutyTransactions
- PD:PD:PagerDutyConfig

Every inbound and outbound transaction flows through the Transactions form. The Request Type and the Action field indicate the transaction which has been captured. Each transaction will use certain fields as part of the process.



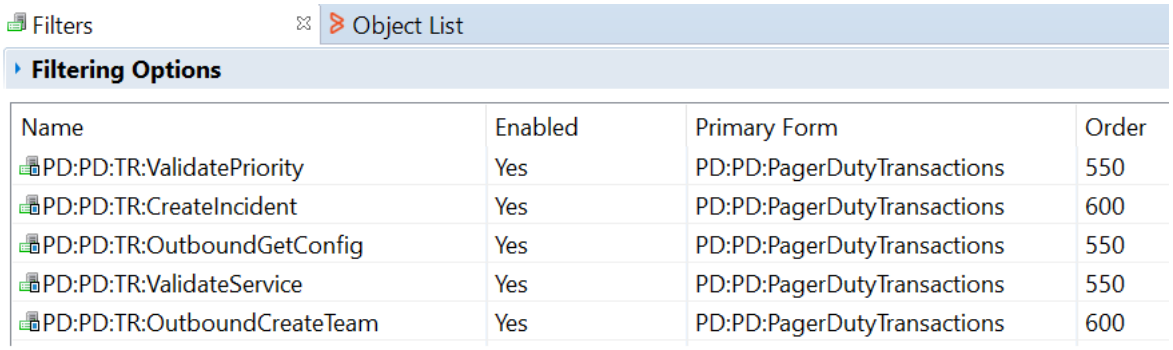
The Config form holds the configuration needed to run the integration. This includes the API Key, the URL as well as information such as the Priority/Group Mapping and the trigger criteria for the outbound transactions. A .arx is provided with sample data for the config.

The detail around which fields are used and which config entries are required are detailed in the function section of this document.

5.2 Filters

All of the workflow is executed via filters. This is to ensure compatibility with Smart IT and also to control the flow via the transaction form. All workflow is prefixed as per the BMC standards.

All PagerDuty specific workflow firing from the PagerDuty forms is setup as PD:PD.



Name	Enabled	Primary Form	Order
PD:PD:TR:ValidatePriority	Yes	PD:PD:PagerDutyTransactions	550
PD:PD:TR:CreateIncident	Yes	PD:PD:PagerDutyTransactions	600
PD:PD:TR:OutboundGetConfig	Yes	PD:PD:PagerDutyTransactions	550
PD:PD:TR:ValidateService	Yes	PD:PD:PagerDutyTransactions	550
PD:PD:TR:OutboundCreateTeam	Yes	PD:PD:PagerDutyTransactions	600

For workflow firing on OOTB forms, the format is always PD:<OOTB Code>. For example PD:HPD.

The name of the filter dictates what it does. This is broken down as follows :

TR: - Transaction – eg: PD:PD:TR

Outbound – Outbound Flow – eg: PD:PD:TR:OutboundCreateIncident

Validate – Data or internal validation eg: PD:PD:TR:ValidateService

<action><type> - For Inbound communications eg: PD:PD:TR:CreateIncident

Each filter is listed within the functions section, underneath which functionality it provides and how it works.

5.2.1 Customisations

For customers who have changed or use bespoke Incident forms, should change the mappings within the filters firing between the Incident form and the Transaction form. Therefore keeping the integration inbound and outbound separate from the OOTB forms.

For example, if you use a custom Service field on Incident, edit the filter PD:HPD:HPD:CreateIncident and change the Service Mapping to the bespoke field.

5.2.2 Non PagerDuty Form Changes

As detailed within the installation section the Non PagerDuty form changes have been kept to a minimum. They use display only fields on Helpdesk, Business Services, Support Group and Work Info. The field in all cases is called z1D Action_PD. It has a common field ID.

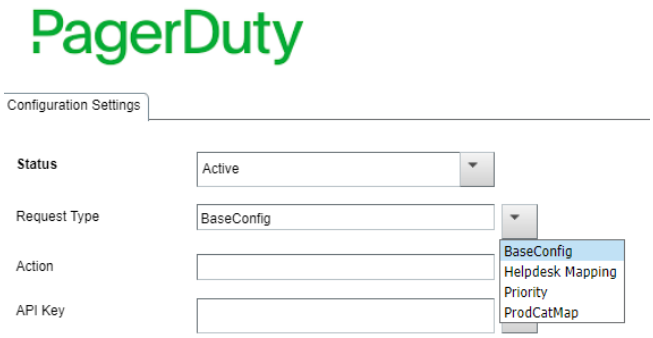
All workflow touching OOTB forms follows the naming conventions mentioned above.

5.3 Configuration Settings/Options

All of the configuration settings are held within the Config form described above. There are four different types of Request Type stored. These are :

- BaseConfig
- Helpdesk Mapping
- Priority
- ProdCatMap

The Status field indicates if the config is 'Active' or not. Only entries in the 'Active' state are used by the integration.

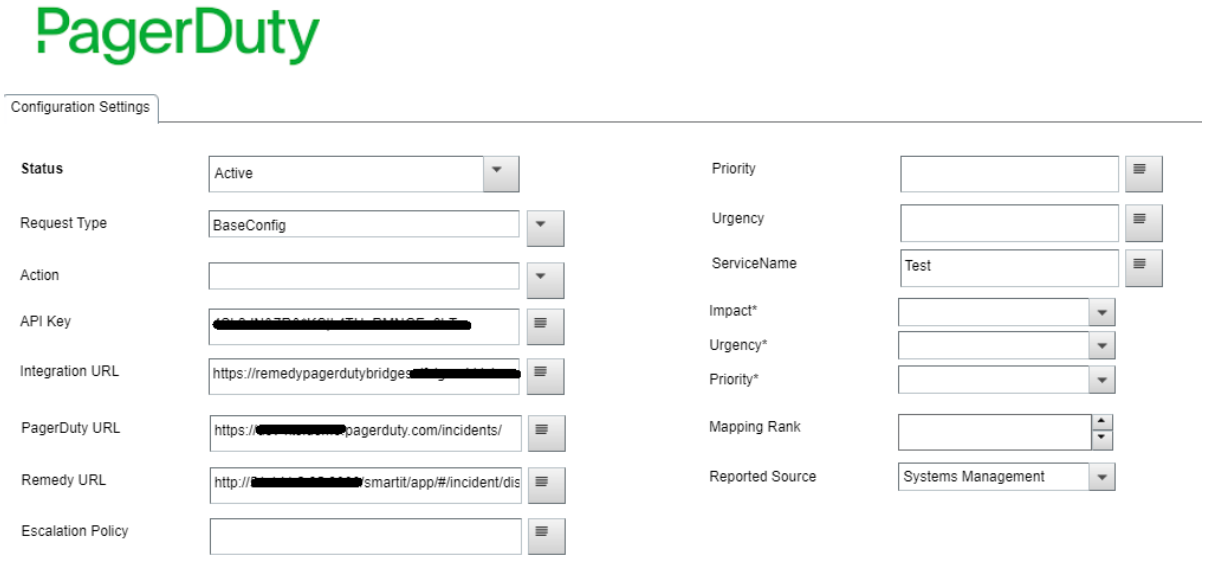


5.3.1 Base Config

The Base Config is mandatory for the integration. It contains the details to connect to the Pagerduty integration and the default values.

For the integration to work the API Key, Integration URL are required. These are the API Key to access PagerDuty and URL for the integration component.

The PagerDuty URL and the Remedy URL are the base URL's needed to write the URL for the ticket to the work logs on submit. These are effectively the URL prefix for the ticket's. Please note the direct Remedy URL will depend if you are using Smart IT or Midtier and the version of Smart IT. Please refer to the Remedy documentation on direct URL links.



5.3.1.1 Base Config Options

You can also specify defaults for the integration. These are :

- Escalation Policy
- ServiceName
- Reported Source

The Escalation Policy will be sent to Pagerduty and will override any Service Escalation Policy.

ServiceName will be used if you wish to specify the SAME service within Pagerduty, this will only be used if no corresponding ProdCatMap exists and/or no Service is sent from the originating Incident within Helix.

The Reported Source field is used when a case is created within Helix. This will be sent to the Incident Create form as oppose to any default set.

5.3.2 Helpdesk Mapping

You can have one or more Helpdesk Mapping entries. These are used to determine if the Incident raised should be sent to Pagerduty. The entries can be ranked by using the Mapping Ranking field in Ascending Order.

The mapping can be any combination of the follow fields :

- Assigned Group
- Service CI
- Operation Categorisation
- Product Categorisation
- Priority

The filter works in the same way as the out of the box Assignment rules. The filter is - PD:HPD:HPD:PagerDuty_Mapping_Lookup. The condition is outlined below :

('Status' = "Active") AND ('Request Type' = "Helpdesk Mapping") AND ('Action' = "Create") AND (('Assigned Group' = $Assigned Group$) OR ('Assigned Group' = $NULL$)) AND (($ServiceCI$ = 'ServiceName_HPD') OR ('ServiceName_HPD' = $NULL$)) AND (($Categorization Tier 1$ = 'Categorization Tier 1') OR ('Categorization Tier 1' = $NULL$)) AND (($Categorization Tier 2$ = 'Categorization Tier 2') OR ('Categorization Tier 2' = $NULL$)) AND (($Categorization Tier 3$ = 'Categorization Tier 2') OR ('Categorization Tier 2' = $NULL$)) AND (($Priority$ = 'Priority_HPD') OR ('Priority_HPD' = $NULL$)) AND (($Product Categorization Tier 1$ = 'Product Categorization Tier 1') OR ('Product Categorization Tier 1' = $NULL$)) AND (($Product Categorization Tier 2$ = 'Product Categorization Tier 2') OR ('Product Categorization Tier 2' = $NULL$)) AND (($Product Categorization Tier 3$ = 'Product Categorization Tier 3') OR ('Product Categorization Tier 3' = $NULL$))

The filter will query the Config Form for the entries in rank order.

Examples :

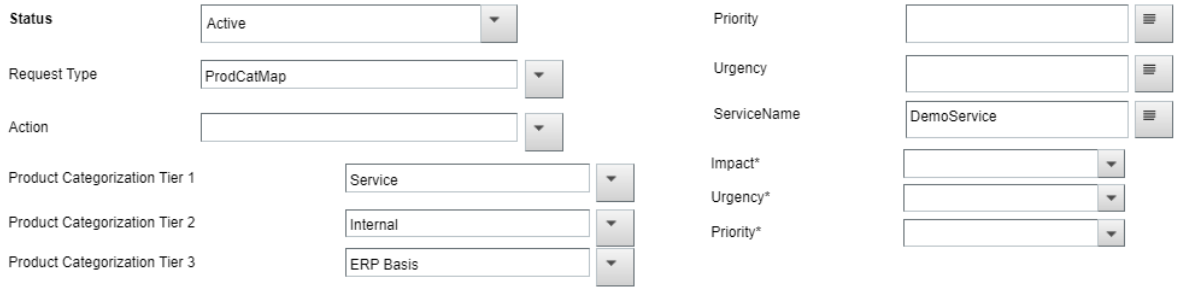
If you wanted to sent entries to Pagerduty for three teams you'd need three Help Desk mapping entries, one for each team. You could then also specify a certain priority for one team on the same entry or a certain Service/Op or Prod cat combination.

5.3.3 Priority

The priority entries are used to map the Priority and Urgency stored within PagerDuty with the Impact, Urgency and Priority in Helix. You need one entry for each map required. Ie: if PagerDuty is setup with P1-P5 you would need five entries, mapped to the 4 corresponding combinations in Helix.

5.3.4 Prod Cat Map

The Prod Cat Map can be used to map a set of Product Categories to a PagerDuty Service. This can be used if the Services between the two environments are not in sync, or Services are not used in Helix. You select the Prod Categories and then enter the ServiceName. You can have multiple entries in this form but only one for each Prod Cat combination.



6 Functions

6.1 Configuration & Service Provisioning

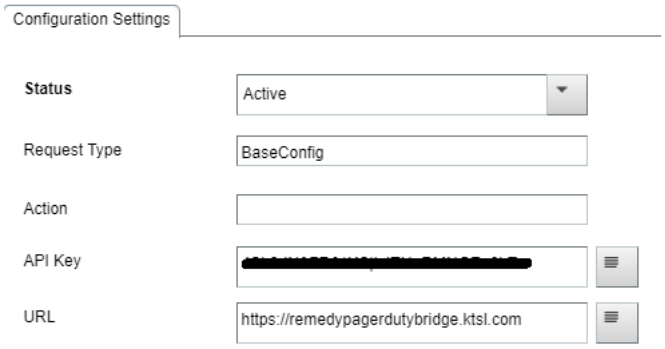
6.1.1 Authentication

The first step in the API is to get the ApiKey. This is retrieved via the API call and written to the configuration within Helix and stored within the application. The ApiKey is then used as the authorisation in the subsequent API requests. The API key is returned as a header parameter.

Security Scheme Type API Key

Header parameter name: ApiKey

Within Helix these are stored in the config form alongside the API URL. The Request Type is BaseConfig. Only the 'Active' value is used. If multiple 'Active' configs exist the first one will be used by the workflow.



For all subsequent Helix workflow calls that are outbound the API Key is read into the Transaction form by this filter :

PD:PD:TR:OutboundGetConfig

6.1.2 Get the Pager Duty config settings

This API call gets the PagerDuty config settings required by the application which are stored within the cloud instance. These include the credentials, URL and responder/company settings for the PagerDuty instance. There is no corresponding call in Helix or workflow.

6.1.2.1 Responses

200 Returns the Pager Duty config values

401 Unauthorized

500 Server Error

get/Config/pagerDutyConfig

6.1.3 Get the Remedy config settings

This API call gets the Helix config settings required by the application which are stored within the cloud instance. These include the credentials and URL endpoint. There is no corresponding call in Helix or workflow.

6.1.3.1 Responses

200 Returns the Remedy config values

401 Unauthorized

500 Server Error

get/Config/remedyConfig

6.1.4 Get details of the Pager Duty webhook subscription

Returns the details of the PagerDuty subscription. This is managed per instance. There is no corresponding call in Helix or workflow.

6.1.4.1 Responses

200 Returns details of the webhook subscription

401 Unauthorized

500 Server Error

get/Config/webhookSubscription

6.1.5 Refreshes the Pager Duty webhook subscription

During the running of the integration the subscription will need to be refreshed. This call refreshes the subscription in order for the integration to continue. It points PagerDuty at the URL specified in the PagerDuty config WebhookUrl property. There is no corresponding call in Helix or workflow.

6.1.5.1 Responses

200 A new webhook subscription was generated

401 Unauthorized

500 Server Error

post/Config/refreshWebhookSubscription

6.1.6 The API endpoint for Pager Duty webhooks

Simply provides the API endpoint for all PagerDuty webhook calls. There is no corresponding call in Helix or workflow.

6.1.6.1 Responses

200 Success

post/Webhook/trigger

6.2 Map Service CI

This is an outbound call to create a Service within PagerDuty from Helix. It's initiated when a new Business Service is created within Helix and set to Deployed. This is called by PD:AST:CreateBusinessService filter which pushes to the transaction form. From here the integration call is invoked from the transactions form by the following filter :

PD:PD:TR:OutboundCreateService

The transactions form then calls the API detailed below.

6.2.1 Create a service in Pager Duty from Helix

6.2.1.1 Request Body schema

Requires details of the Service to create, if the service name already exists a duplicate will not be created.

name string

requesterEmail string

6.2.2 Responses

200 Success

401 Unauthorized

500 Server Error

post/RemedyBusinessServices/provision

6.2.3 Request sample payload

```
{"name": "string",
"requesterEmail": "string"
}
```

6.3 Provisioning Users, Teams from Helix

This step allows you to create Teams within PagerDuty from Helix. When a new support group is created within Helix, you need to create a mapping in the Config form to invoke the integration. This indicates that the support group is being leveraged by the integration. The config form calls the filter PD:PD:CF:CreateTeam. This will retrieve information from the Support Group to User join within Helix and send the information to PagerDuty via the API's. This is invoked by the filter PD:PD:TR:OutboundCreateTeam.

The transactions form then calls the following API.

6.3.1 Create a team in Pager Duty from Helix

6.3.1.1 Request Body schema

requesterEmail string

name string

description string

users Array of strings

6.3.1.2 Responses

200 Success

post/RemedyTeams/provision

6.3.1.3 Request sample payload


```
{"requesterEmail":"string",
"name":"string",
"description":"string",
"users":["string"]
}
```


6.3.2 Create a user in Pager Duty to match Helix

6.3.2.1 Request Body schema

name string

email string

role string

jobTitlestring

timeZone string

description string

color string

requesterEmail string

6.3.2.2 Responses

200 Success

401 Unauthorized

500 Server Error

post/RemedyUsers/provision

6.3.2.3 Request samples payload


```
{"name":"string",
"email":"string",
"role":"string",
"jobTitle":"string",
"timeZone":"string",
"description":"string",
"color":"string",
"requesterEmail":"string"
}
```


6.4 Trigger Helix Incidents to Pagerduty

This is an outbound trigger from Helix to Pagerduty for Incident Creation. The creation is invoked when a mapping within the config form matches the criteria on Help Desk. For example the support group the ticket is assigned to matches an assignment mapping. The

available mappings are Support Group, Operational Categorisation and Service CI. The mapping is checked by the filter PD:HPD:HPD:PagerDuty_Mapping_Lookup. If the mapping criteria is met then this filter fires PD:HPD:HPD:CreateIncident. This creates the entry in the transaction form. The transaction form fires the filter PD:PD:TR:OutboundCreateIncident. Once the entry is created within PagerDuty, the PagerDuty ID is sent back to Helix. The return is then pushed to the Helpdesk Incident via this filter PD:PD:TR:UpdateVendorID.

6.4.1 Request Body schema

requesterEmail	string
remedyIncidentId	string
title	string
serviceName	string
priority	string
details	string

6.4.2 Responses

200 Success

post/RemedyIncidents

6.4.3 Request sample payload

```
{"requesterEmail":"string",
"remedyIncidentId":"string",
"title":"string",
"serviceName":"string",
"priority":"string",
"details":"string"
}
```

6.4.4 Set Vendor ID in Remedy when initiated in Helix

This is a sample payload for the API call into the Transactions form.

{"values":

{"Request Type":"PDIncident","Action":"UpdateVendorId","Id":"123241","VendorID":"PXDM0FK"}

}

6.5 Trigger PagerDuty to Helix

This call creates the Incident in Helix when created within Pagerduty. The API creates the entry within the transaction form and this calls the filter PD:PD:TR:CreateIncident. This pushes to the Incident Create form which generates the Incident.

6.5.1 Sample Payload

The below is a sample inbound payload to the Transaction form.

{"values":

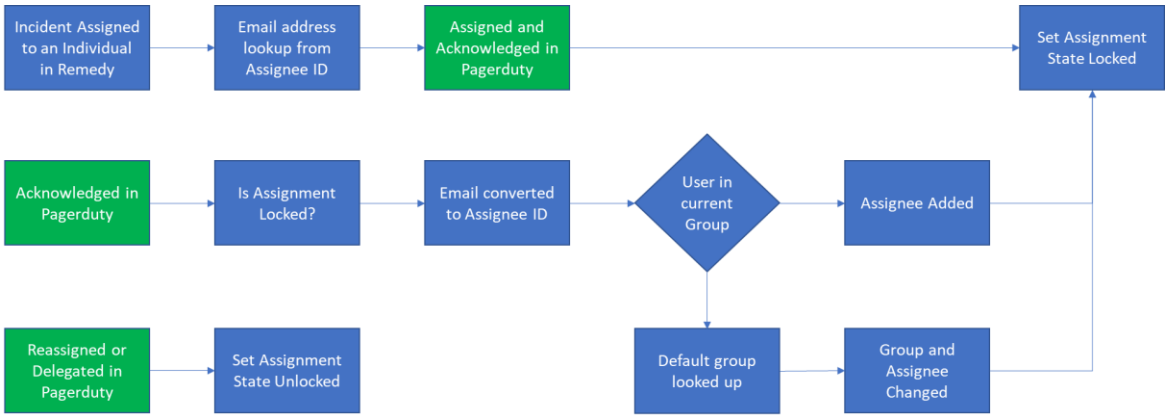
{"Request Type":"PDIncident","Action":"Create","VendorID":"PI70VM2","inp_action":"triggered","Title":"Another Test","Device ID":"[#68] Example","Assignee":null,"Assigned Group":null,"Priority":"low","Urgency":"high","ServiceName":"X1"}

}

6.6 Acknowledge/Assignment Rules

The sync of assignment and acknowledge is controlled by a 'state'. Remedy will take priority over Pagerduty for assignment. Within PagerDuty the initial acknowledgement will set the Remedy assignee, and lock the 'state'. Subsequent Acknowledgment will then not change the Remedy assignee, until a re-assign or delegate is issued within Pagerduty. This will then change the 'state' in Remedy to unlocked and allow the next Acknowledgement in Pagerduty to update Remedy again. Any assignment changes in Remedy will always change the assignee and auto acknowledge the Pagerduty Incident.

6.6.1 Acknowledge/Assignment Flow



In Assignment lock state future acknowledgements do not change assignment in Remedy. A re-assign or delegation must happen first. This sets the state to unlocked.

6.6.2 Sync Acknowledgement (PD->R)

When a ticket is acknowledged within PagerDuty this writes the acknowledgement into the Work Info on the Incident. The API sends the acknowledgement into the transaction form. The following filter fires to get the Incident detail in order to submit the Work Info entry PD:PD:TR:WorkNote_GetValues. Once that has fired PD:PD:TR:Acknowledge is called to create the Work Info.

6.6.2.1 Sample Payload

{"values":

{"Request Type":"PDIncident","Action":"Acknowledge","VendorID":"PYEGCBB"}

}

6.6.3 Assign a Team to an incident in Pager Duty

When the assignment changes within Helix where a PagerDuty ID is assigned it will update the assignment within PagerDuty. Note this filter should be disabled where Escalations and Assignment is being managed within PagerDuty. The filter is PD:HPD:HPD:AssignmentChange. This calls PD:PD:TR:OutboundAssignment on the transaction form which calls the API.

6.6.3.1 path Parameters

pagerDutyIncidentId required string

6.6.3.2 Request Body schema

requesterEmail string -The email of the Remedy user making the request

name required string -The team's name

6.6.3.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/assignteam

6.6.3.4 Request sample payload

```
{"requesterEmail":"string",
"name":"string"
}
```

6.6.4 Assign a User to an Incident in Pager Duty

When the assigned user changes within Helix where a PagerDuty ID is assigned it will update the assignment within PagerDuty. Note this filter should be disabled where Escalations and Assignment is being managed within PagerDuty. The filter is PD:HPD:HPD:AssigneeChange. This calls PD:PD:TR:OutboundAssignee on the transaction form which calls the API.

6.6.4.1 path Parameters

pagerDutyIncidentId required string

6.6.4.2 Request Body schema

requesterEmail string -The email of the Remedy user doing the assignment

assignUserEmail string -The email of the Remedy user being assigned

6.6.4.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/assignuser

6.6.4.4 Request sample payload

```
{"requesterEmail":"string",
"assignUserEmail":"string"
}
```

6.7 Sync Resolve (R->PD & PD->R)

This allows a bi-directional sync if a ticket is resolved.

6.7.1 Helix to Pagerduty

On the outbound call, if the PagerDuty ID is on the Helpdesk form then this pushes the resolution update to the transaction form via PD:HPD:HPD:ResolveIncident. The transaction form then calls the filter PD:PD:TR:OutboundResolve. This invokes the API.

6.7.1.1 path Parameters

pagerDutyIncidentId required string

6.7.1.2 Request Body schema:

resolveReason string - A description of why/how the incident was resolved

requesterEmail string - The Remedy user who resolved the incident

6.7.1.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/resolved

6.7.1.4 Request sample payload

```
{"resolveReason":"string",
"requesterEmail":"string"
}
```

6.7.2 Pagerduty – Helix

When the resolution is invoked from PagerDuty a note API and a resolve API are called. From within the transaction form this will call the filter PD:PD:TR:ResolveIncident and PD:PD:TR:AddResolutionNote. Note the resolution note will use the generic note mapping filter.

6.7.3 Sample Payload

{"values":
{"Request Type":"PDIncident","Action":"AddNote","VendorID":"PI70VM2","Note":"Resolution Note: Resolved"}
}
{"values":
{"Request Type":"PDIncident","Action":"Resolve","VendorID":"PI70VM2"}
}

6.8 Sync Notes, Status Updates into Work Items

This section covers bi-directional note and status updates between Pagerduty and Helix.

6.8.1 Note added in Helix

When a Work Info is added within Helix it will be sent to PagerDuty if the parent Incident has a PagerDuty ID associated with it and it's of the type Paging System. The filter PD:HPD:HPD:WorkInfoMapping will check if the parent is related and then PD:HPD:HPD:WorkInfoCreate will push to the transaction form. On the transaction form, filter PD:PD:TR:OutboundAddNotes will call the API.

6.8.1.1 path Parameters

pagerDutyIncidentId required string

6.8.1.2 Request Body schema:

requesterEmail string - The email of the Remedy user adding the note

note string - The content of the note

6.8.1.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/notes

6.8.1.4 Request sample payload

```
{"requesterEmail":"string",
"note":"string"
}
```

6.8.2 Status update has been added in Helix

When a status change has happened within Helix this will update PagerDuty. Note that PagerDuty terminology for this is a Status Update. This will fire from the Work Info note added within Helix as oppose to directly from the Incident. The same mapping filter is used but then PD:HPD:HPD:WorkInfoCreateStatusUpdate is called to push to the transaction form. PD:PD:TR:OutboundStatusUpdate on the transaction form to invokes the API.

6.8.2.1 path Parameters

pagerDutyIncidentId

required string

6.8.2.2 Request Body schema

requesterEmail string -The email of the Remedy user adding the status update

message string -The status update message

6.8.2.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/statusupdates

6.8.2.4 Request sample payload

```
{"requesterEmail":"string",
"message":"string"
}
```

6.8.3 Note Update from PagerDuty

When a note comes in from PagerDuty it will call the same generic filter as Acknowledge and Escalate to get the values PD:PD:TR:WorkNote_GetValues. Then the filter PD:PD:TR:Note will add the Work Info.

6.8.3.1 Sample Payload

{"values":

{"Request Type":"PDIncident","Action":"AddNote","VendorID":"PYEGCBB","Note":"New Note"}

}

6.8.4 Status Update from PagerDuty

When a note comes in from PagerDuty it will call the same generic filter as above to get the values PD:PD:TR:WorkNote_GetValues. Then the filter PD:PD:TR:StatusUpdate will add the Work Info.

6.8.4.1 Sample Payload

{"values":

{"Request Type":"PDIncident","Action":"AddStatusUpdate","VendorID":"PI70VM2","StatusUpdate":"test update"}

}

6.9 PagerDuty Escalation

This will fire when an Escalation is invoked from within PagerDuty. It will create a transaction entry which will use the generic Work Info values filter. The filter PD:PD:TR:Escalate then creates the Work Info.

6.9.1 Sample Payload

{"values":

{"Request Type":"PDIncident","Action":"Escalate","VendorID":"PI70VM2"}

}

6.10 Add Responders

This updates Helix when a responder is added within PagerDuty. This will update the Work Info within Remedy rather than re-assign the Incident as you can have multiple responders within PagerDuty. If required this can be changed within the transaction filter. The API will create the entry in the transaction form which calls the filter PD:PD:TR:AddResponder.

6.10.1 Sample Payload

{"values":

{"Request Type":"PDIncident","Action":"AddNote","VendorID":"PI70VM2","Name":"Ian Walsh"}

}

6.11 Run Response Plays

This allows Helix to invoke a Response Play within PagerDuty. This is fired when a Work Info note is created with the corresponding Respond Play named. This filter fires to create the transaction entry PD:HPD:HPD:RunResponsePlay. Then this filter fires to call the API PD:PD:TR:OutboundRunResponsePlay.

6.11.1 path Parameters

pagerDutyIncidentId required string

6.11.2 Request Body schema:

responsePlayId string -The id of the reponse play to run

requesterEmail string -The email of the Remedy user requesting the response play

6.11.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/runresponseplay

6.11.4 Request sample payload

```
{"responsePlayId":"string",
"requesterEmail":"string"
}
```

6.12 Sync Priority

This section covers bi-directional priority updates between PagerDuty and Helix.

6.12.1 Priority Changed in Helix

When the priority is changed within Helix and the Vendor ID is not NULL then the PD:HPD:HPD:PriorityIncident filter will push to the transaction form. On the transaction form, the priority mapping will be read in first via the filter PD:PD:TR:OutboundGetIncPriority. Then the filter PD:PD:TR:OutboundPriority will call the API.

6.12.1.1 path Parameters

pagerDutyIncidentId required string

6.12.1.2 Request Body schema:

priority string - priority mapping

urgency string - urgency mapping

6.12.1.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/notes

6.12.1.4 Request sample payload

```
{"priority":"string",
"urgency":"string"
}
```

6.12.2 Priority Changed in PagerDuty

When a status change has happened within PagerDuty this will update Helix. This will create a PriorityUpdate entry in the transactions form. The priority mapping filter will then run PD:PD:TR:ValidatePriority , this will then run PD:PD:TR:PriorityIncident to update the Incident record.

6.12.2.1 path Parameters

pagerDutyIncidentId required string

pagerDutyId required string

6.12.2.2 Request Body schema:

priority string - priority mapping

urgency string - urgency mapping

6.12.2.3 Responses

200 Success

post/RemedyIncidents/{pagerDutyIncidentId}/notes

6.12.2.4 Request sample payload

```
{"priority":"string",
"urgency":"string"
}
```