How to Sync Time-related Information between ServiceNow and Jira

Table of contents

This question has been raised on ServiceNow community

This page details the answer to this question.

1. Question

We have the following workflow:

- An incident is raised on ServiceNow.
- This incident escalates towards Jira where an Epic is created.
- The Epic is broken down into stories. Every story has a time estimate and the actual time spent information.

The question is:

Is it possible to synchronize this time-related information back to the incident, so that we can keep track of the budget consumed by the development activities?

2. Answer

2.1. Overview

This use case is an advanced synchronization case and needs quite some explanation. The overall flow is as follows:

- The incident is escalated to Jira using an Exalate Trigger.
- The Exalate on Jira creates an epic.

- This epic is broken down into stories.
- Whenever a story is created and/ or modified, sync is triggered on the parent epic.
- During the sync from the Epic to the incident, the time tracking-related information is tallied and included in the message to the incident.
- This information is included in custom fields on the incident.

2.2. The details

2.2.1. The incident is escalated to Jira

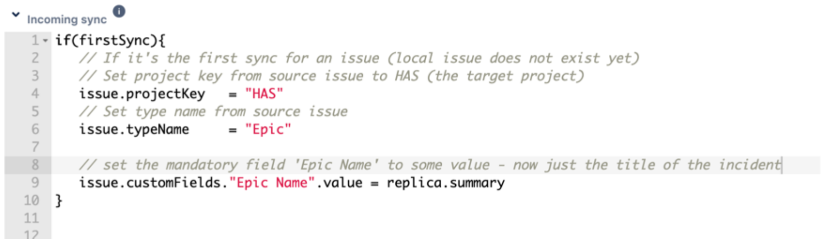
In Exalate for ServiceNow, you can define a trigger - which will regularly check (every 20 secs) if an incident should be synchronized.

In this case, an additional state has been added, such that the user has an easy way to send over the incident.

The incident gets wrapped into a message, and the message is sent over to Jira.

2.2.2. The Exalate on Jira creates an epic

The Exalate for Jira Server receives the message, finds out if this is the first time this incident is being synced, and processes the following code:



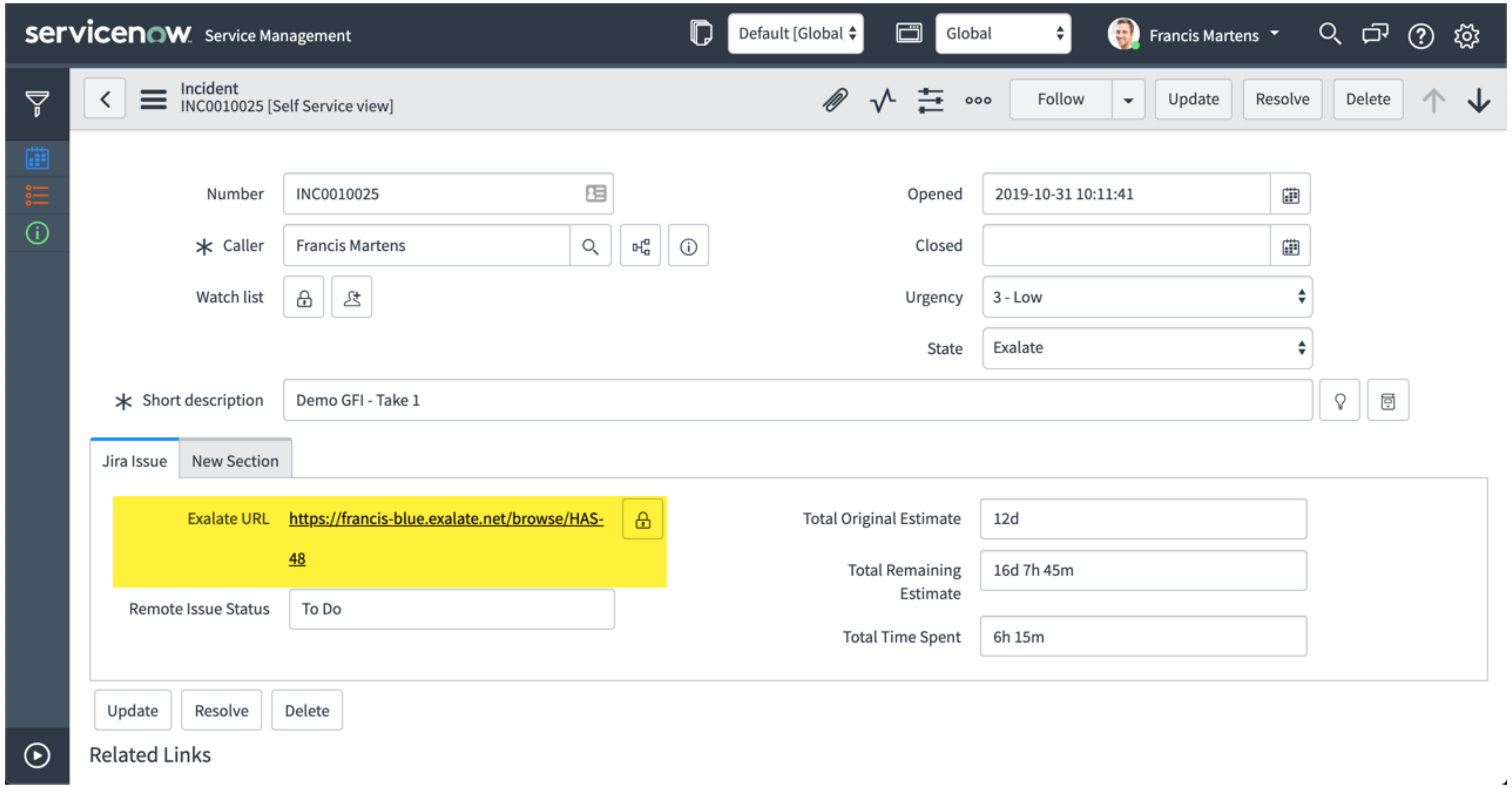
```
if(firstSync){
    // If it's the first sync for an issue (local issue does not exist yet)
    // Set project key from source issue to HAS (the target project)
    issue.projectKey    = "HAS"
    // Set type name from source issue
    issue.typeName      = "Epic"

    // set the mandatory field 'Epic Name' to some value - now just the title of the incident
    issue.customFields."Epic Name".value = replica.summary
}
```

This will

- Create the epic.
- fill in the epic name custom field (which happens to be mandatory for Epics).
- Create a synchronization relationship between the incident and the epic.

A link is also added to the incident.

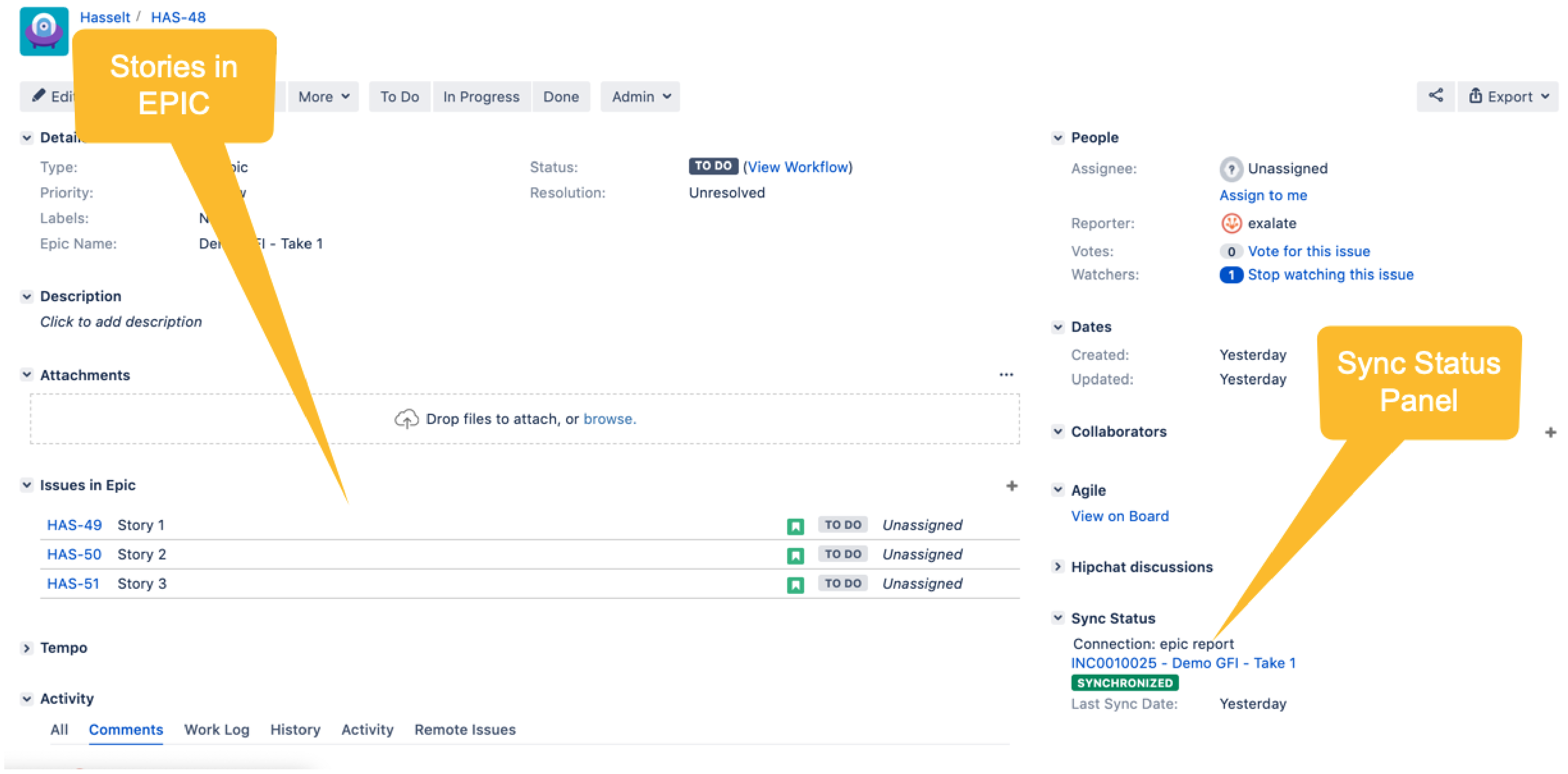


Number	INC0010025	Opened	2019-10-31 10:11:41
Caller	Francis Martens	Closed	
Watch list		Urgency	3 - Low
		State	Exalate
Short description	Demo GFI - Take 1		

Exalate URL	https://francis-blue.exalate.net/browse/HAS-48	Total Original Estimate	12d
		Total Remaining Estimate	16d 7h 45m
Remote Issue Status	To Do	Total Time Spent	6h 15m

2.2.3. This epic is broken down into stories

This is something that the development team does and results in a detailed 'mini-project-plan'.



Details

Type: Epic
Priority:
Labels:
Epic Name: De... I - Take 1

Status: TO DO (View Workflow)
Resolution: Unresolved

Description

Click to add description

Attachments

Drop files to attach, or browse.

Issues in Epic

HAS-49	Story 1	TO DO	*Unassigned*
HAS-50	Story 2	TO DO	*Unassigned*
HAS-51	Story 3	TO DO	*Unassigned*

Tempo

Activity

All Comments Work Log History Activity Remote Issues

People

Assignee: Unassigned
Assign to me
Reporter: exalate
Votes: 0 Vote for this issue
Watchers: 1 Stop watching this issue

Dates

Created: Yesterday
Updated: Yesterday

Collaborators

Agile

View on Board

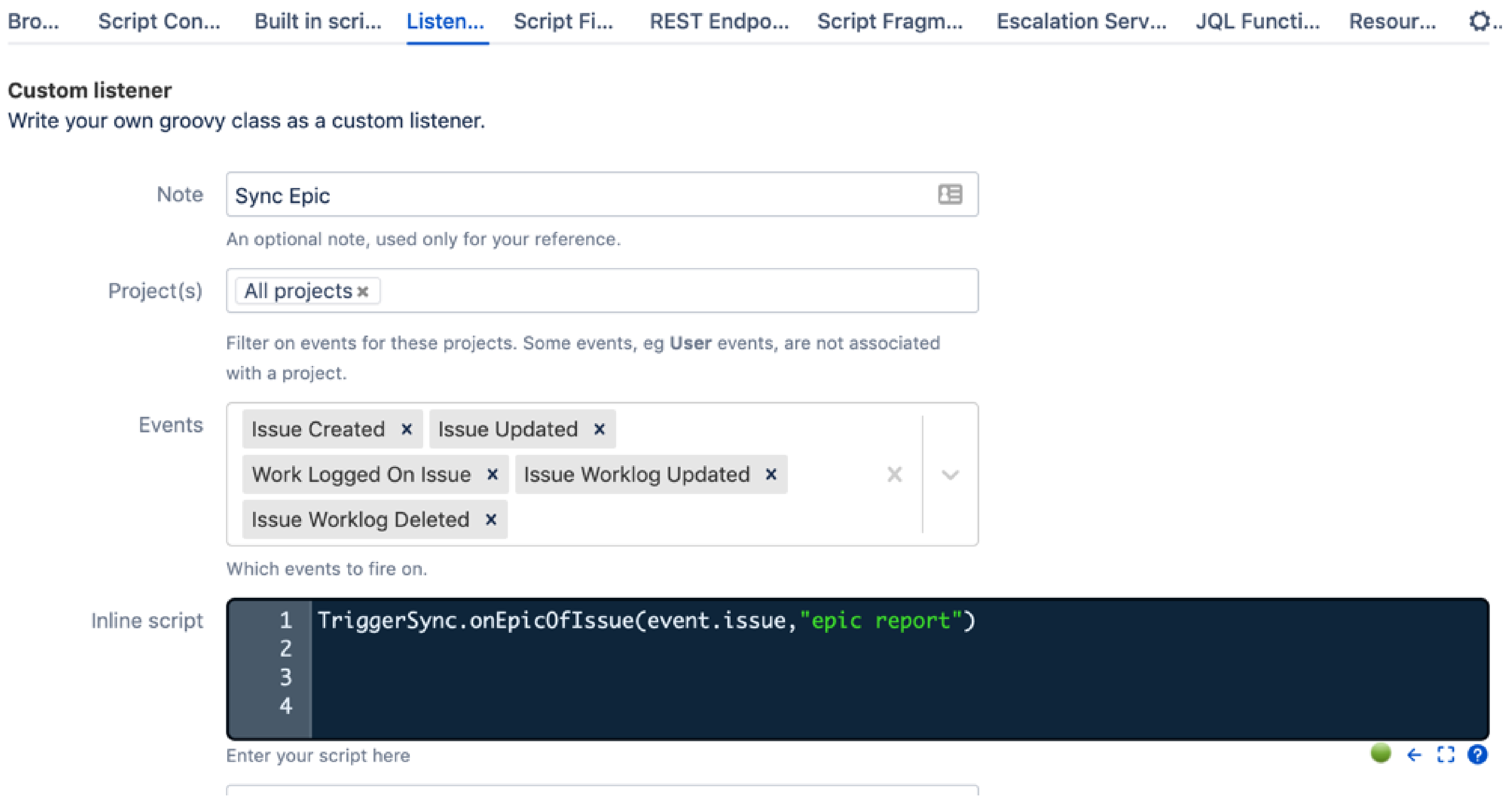
Hipchat discussions

Sync Status

Connection: epic report
INC0010025 - Demo GFI - Take 1
SYNCHRONIZED
Last Sync Date: Yesterday

2.2.4. Whenever a story is created and/ or modified, sync is triggered on the parent epic

Using a 'script listener', update events on stories which are escalated into the sync of the epic.



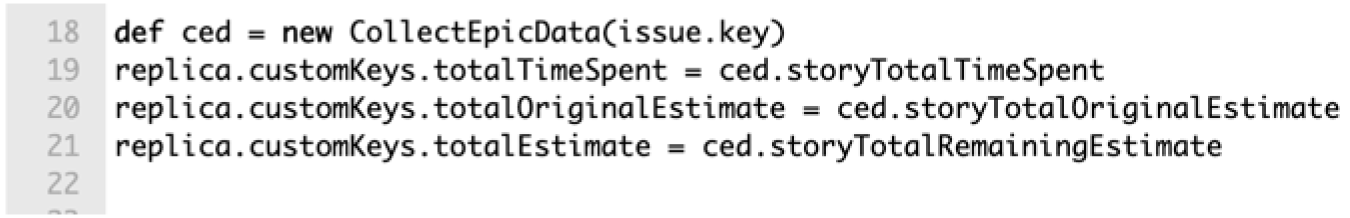
Custom listener
Write your own groovy class as a custom listener.

Note	Sync Epic
	An optional note, used only for your reference.
Project(s)	All projects
	Filter on events for these projects. Some events, eg **User** events, are not associated with a project.
Events	Issue Created, Issue Updated, Work Logged On Issue, Issue Worklog Updated, Issue Worklog Deleted
	Which events to fire on.
Inline script	(see below)
	Enter your script here

```
TriggerSync.onEpicOfIssue(event.issue,"epic report")
```

2.2.5. During the sync from the Epic to the incident, the time tracking related information is tallied and included in the message to the incident

Whenever an epic is synced, all the relevant information is collected from the underlying stories (and optionally subtasks), using the following code

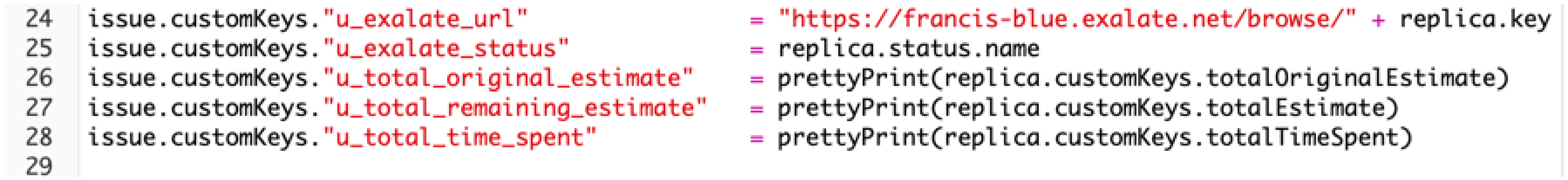


```
18  def ced = new CollectEpicData(issue.key)
19  replica.customKeys.totalTimeSpent = ced.storyTotalTimeSpent
20  replica.customKeys.totalOriginalEstimate = ced.storyTotalOriginalEstimate
21  replica.customKeys.totalEstimate = ced.storyTotalRemainingEstimate
22
```

- Line 18 is calling an externalized script which contains the logic to tally all relevant information.
- Line 19 - Line 21 include this information into the message sent to Exalate for ServiceNow.

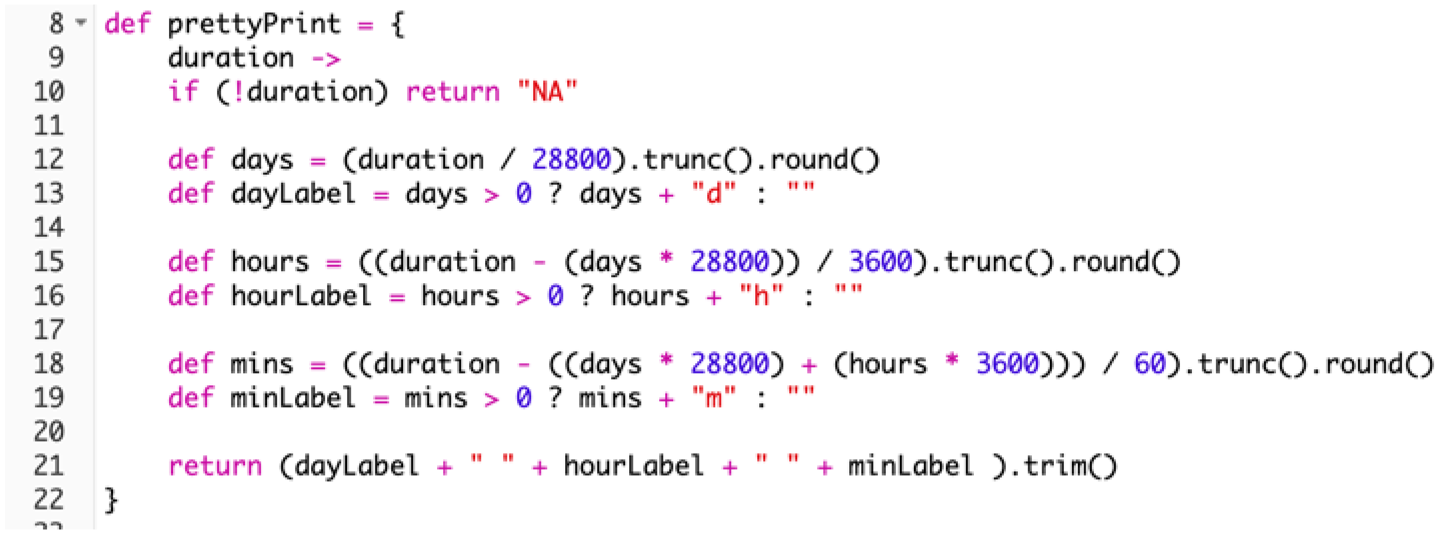
2.2.6. This information is included in custom fields on the incident

The code to update the ServiceNow incident is:



```
issue.customKeys."u_exalate_url"                = "https://francis-blue.exalate.net/browse/" + replica.key
issue.customKeys."u_exalate_status"             = replica.status.name
issue.customKeys."u_total_original_estimate"    = prettyPrint(replica.customKeys.totalOriginalEstimate)
issue.customKeys."u_total_remaining_estimate"   = prettyPrint(replica.customKeys.totalEstimate)
issue.customKeys."u_total_time_spent"           = prettyPrint(replica.customKeys.totalTimeSpent)
```

The prettyPrint method is a custom method (included in the script) which transforms the duration into a pretty print format (expressing time in days/hours/minutes).



```
def prettyPrint = {
    duration ->
    if (!duration) return "NA"

    def days = (duration / 28800).trunc().round()
    def dayLabel = days > 0 ? days + "d" : ""

    def hours = ((duration - (days * 28800)) / 3600).trunc().round()
    def hourLabel = hours > 0 ? hours + "h" : ""

    def mins = ((duration - ((days * 28800) + (hours * 3600))) / 60).trunc().round()
    def minLabel = mins > 0 ? mins + "m" : ""

    return (dayLabel + " " + hourLabel + " " + minLabel ).trim()
}
```

2.3. Getting access

The whole configuration is available and can be requested through our support channel.

© Exalate 2024