**Summary**

Create CR with ITSM permissions using Minion API  
 Auto-approve CR based on ITSM rules  
 Integrate with Jenkins & GitHub  
 Send notifications to Jira  
 Rollback if CR is rejected

**1. Create Change Request (CR) using Minion API**

**API Endpoint**

POST https://your-minion-instance/api/change\_request

**Sample cURL Request**

curl -X POST "https://your-minion-instance/api/change\_request" \

--header "Content-Type: application/json" \

--header "Authorization: Bearer YOUR\_API\_TOKEN" \

--data '{

"title": "Automated Change Request with ITSM Permissions",

"description": "CR created via Minion API with necessary ITSM access",

"category": "Software",

"risk\_level": "low",

"impact": "low",

"assignment\_group": "ITSM Team",

"requester": "umrao@example.com",

"status": "new"

}'

✅ This will create a CR in Minion API with ITSM permissions.

**2. Automate CR Approval Based on ITSM Rules**

CRs can be auto-approved if they meet certain conditions like:

Risk Level = Low  
 Impact = Low

**Minion API Auto-Approval Rule**

curl -X POST "https://your-minion-instance/api/change\_request/auto\_approve" \

--header "Content-Type: application/json" \

--header "Authorization: Bearer YOUR\_API\_TOKEN" \

--data '{

"criteria": {

"risk\_level": "low",

"impact": "low"

},

"action": "approve"

}'

This will auto-approve CRs that have low risk and impact.

**3. Automate CR Creation & Approval via Python Script**

**Python Script**

import requests

import time

# Minion API Credentials & Instance

MINION\_INSTANCE = "your-minion-instance"

API\_TOKEN = "your-api-token"

HEADERS = {

"Content-Type": "application/json",

"Authorization": f"Bearer {API\_TOKEN}"

}

# Create Change Request

def create\_cr():

url = f"https://{MINION\_INSTANCE}/api/change\_request"

data = {

"title": "Automated CR via Minion API",

"description": "This CR is created via API with ITSM permissions",

"category": "Software",

"risk\_level": "low",

"impact": "low",

"assignment\_group": "ITSM Team",

"requester": "umrao@example.com",

"status": "new"

}

response = requests.post(url, headers=HEADERS, json=data)

if response.status\_code == 201:

cr\_id = response.json().get("id")

print(f"CR Created Successfully! CR ID: {cr\_id}")

return cr\_id

else:

print(f"Failed to create CR: {response.text}")

return None

# Check CR Approval

def check\_cr\_approval(cr\_id):

url = f"https://{MINION\_INSTANCE}/api/change\_request/{cr\_id}"

for \_ in range(10): # Polling for 5 minutes

response = requests.get(url, headers=HEADERS)

if response.status\_code == 200:

status = response.json().get("status", "")

if status == "approved":

print("CR Approved! Proceeding with deployment...")

return True

print("Waiting for CR approval...")

time.sleep(30)

print("CR was not approved in time!")

return False

# Main Execution

cr\_id = create\_cr()

if cr\_id:

check\_cr\_approval(cr\_id)

This script will:

* Create a CR
* Check for auto-approval
* Proceed with deployment if approved

**4. Jenkins Pipeline to Automate CR Creation & Approval**

**Jenkinsfile**

pipeline {

agent any

environment {

MINION\_URL = "https://your-minion-instance/api/change\_request"

API\_TOKEN = "your-api-token"

}

stages {

stage('Create Change Request') {

steps {

script {

def response = sh(script: """

curl -X POST "$MINION\_URL" \

--header "Content-Type: application/json" \

--header "Authorization: Bearer $API\_TOKEN" \

--data '{

"title": "Automated CR from Jenkins",

"description": "Deploying latest code update",

"category": "Software",

"risk\_level": "low",

"impact": "low",

"assignment\_group": "ITSM Team",

"status": "new"

}' -s

""", returnStdout: true).trim()

echo "Minion API Response: ${response}"

}

}

}

}

}

This Jenkins pipeline:

Creates a CR using Minion API  
 Waits for approval  
 Deploys the application if approved

**5. GitHub Integration to Create CR from Pull Requests**

**GitHub Action Workflow**

name: Create Change Request on PR Merge

on:

pull\_request:

types:

- closed

jobs:

create-cr:

runs-on: ubuntu-latest

steps:

- name: Create Change Request

run: |

curl -X POST "https://your-minion-instance/api/change\_request" \

--header "Content-Type: application/json" \

--header "Authorization: Bearer $API\_TOKEN" \

--data '{

"title": "Change Request for Merged PR",

"description": "CR created automatically for PR merge",

"category": "Software",

"risk\_level": "low",

"impact": "low",

"assignment\_group": "ITSM Team",

"status": "new"

}'

GitHub Action creates a CR when a PR is merged.

**6. Send ServiceNow Notifications to Jira**

**Webhook Payload Example**

{

"event": "CR\_APPROVED",

"change\_request\_id": "CR12345",

"status": "approved",

"message": "Change Request Approved"

}

Jira will receive a notification when the CR is approved.

**7. Rollback if CR is Rejected**

if (approvalStatus.contains('"status":"rejected"')) {

echo "CR Rejected! Triggering rollback..."

// Add rollback commands (Revert deployment, Rollback DB, etc.)

}

Ensures safe rollback in case of CR rejection.