# Guida all'analisi dei Workflows di Jira

## Introduzione

Questa guida descrive il processo per identificare e analizzare riferimenti a porte specifiche (come la 8080) negli script di Jira, particolarmente utile durante migrazioni o aggiornamenti di infrastruttura.

Il problema dei riferimenti hardcoded a porte o URL si manifesta principalmente in due contesti:

1. **Script Groovy su file system**: Script che risiedono sul file system del server Jira e possono essere modificati con un semplice find & replace tramite script bash.
2. **Script embedded in Jira**: Script incorporati direttamente nei flussi di lavoro (nelle post-function, workflow, ecc.). Per questi è necessaria l'analisi dettagliata che questa guida aiuta a realizzare.

## Cosa verificare:

* verificare prima se gli script sono su file system (più semplice)
* nel caso di script embedded, bisognerà capire come fare.. probabilmente attraverso le API di Jira o strumenti di amministrazione?

Simone ha già esperienza con un processo simile per il passaggio da Prod a Test; quindi, potremmo adattare il suo approccio per questo caso specifico della porta 8080.

## Prerequisiti

* Accesso amministrativo a Jira
* Plugin ScriptRunner installato su Jira
* Python 3.6 o superiore installato sul computer locale
* Script base64\_xml\_decoder.py (fornito a parte)

## Procedura

**Fase 1: Esportazione dei Workflows di Jira**

1. Accedere al pannello di Amministrazione di Jira
2. Navigare su "Flussi di lavoro" nella colonna di sinistra  
   Immagine che contiene testo, schermata, software, Icona del computer

   Descrizione generata automaticamente
3. Selezionare "ScriptRunner" nella tab superiore  
   Immagine che contiene testo, Carattere, numero, linea

   Descrizione generata automaticamente
4. Accedere alla tab "Console" all'interno di ScriptRunner (vedi sempre foto punto 3)
5. Incollare il seguente codice nella console:  
   Immagine che contiene testo, elettronica, schermata, software

   Descrizione generata automaticamente  
   Codice:

import com.atlassian.jira.component.ComponentAccessor

import com.atlassian.jira.workflow.WorkflowUtil

import java.util.zip.ZipOutputStream

import java.util.zip.ZipEntry

def workflowManager = ComponentAccessor.getWorkflowManager()

def baos = new ByteArrayOutputStream()

def zipStream = new ZipOutputStream(baos)

try {

    workflowManager.getWorkflows().each { wf ->

        def safeName = wf.name.replaceAll(/[^\p{Alnum}-\_]/, "\_") + ".xml"

        def xmlContent = WorkflowUtil.convertDescriptorToXML(wf.getDescriptor())

        zipStream.putNextEntry(new ZipEntry(safeName))

        zipStream.write(xmlContent.toString().getBytes("UTF-8"))

        zipStream.closeEntry()

    }

    zipStream.close()

    def zipBytes = baos.toByteArray()

    def fileName = "jira-workflows-${new Date().format('yyyyMMdd-HHmm')}.zip"

    // Crea un link di download

    def downloadLink = """

        <a href='data:application/zip;base64,${zipBytes.encodeBase64().toString()}'

           download='${fileName}'

           style='font-size:16px;padding:10px;background:#3572b0;color:white;text-decoration:none;border-radius:4px;'>

           Download Workflows (${workflowManager.getWorkflows().size()} workflow)

        </a>

    """

    return downloadLink

} catch(Exception e) {

    log.error("Errore durante l'esportazione: ${e.message}", e)

    return "<div style='color:red'>Errore: ${e.message}</div>"

}

1. Cliccare sul pulsante "Run"  
   Immagine che contiene testo, schermata, Carattere, numero

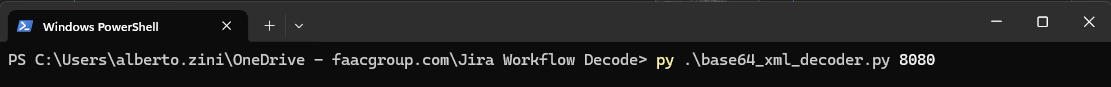
   Descrizione generata automaticamente
2. Apparirà un nuovo pulsante "**Download Workflows ({num\_workflow})**" subito sotto al pulsante "Run" (vedi foto punto 6)
3. Cliccare sul pulsante per scaricare il file ZIP contenente tutti i workflows in formato XML (vedi foto punto 6)

**Fase 2: Preparazione dei file per l'analisi**

1. Estrarre il contenuto del file ZIP scaricato
2. Copiare tutti i file XML estratti nella cartella ***xml*** presente nella stessa directory dello script ***base64\_xml\_decoder.py***

**Fase 3: Decodifica e ricerca nei Workflows**

1. Aprire una finestra di terminale o prompt dei comandi nella cartella dove è stato estratto l’archivio zip contenente lo script  
   Immagine che contiene testo, schermata, software, Software multimediale

   Descrizione generata automaticamente
2. Navigare alla directory contenente lo script **base64\_xml\_decoder.py**
3. Eseguire il comando: ***python base64\_xml\_decoder.py 8080  
   ***

Sostituire "8080" con qualsiasi altro termine di ricerca desiderato (es. nomi di server, URL, altri numeri di porta)

1. Lo script eseguirà le seguenti operazioni:
   * Creerà la directory ***xml-decoded*** se non esiste già
   * Leggerà e decodificherà i file XML dalla cartella ***xml***
   * Salverà le versioni decodificate nella cartella ***xml-decoded***
   * Cercherà il termine specificato (es. "8080") in tutti i file decodificati
   * Genererà un report HTML con i risultati della ricerca

**Fase 4: Analisi dei risultati**

1. Al termine dell'esecuzione, si genererà automaticamente nella cartella dello script, il file ***result.html*** con i risultati della ricerca, visualizzabile tramite browser
2. Il report mostrerà:

* Nome del workflow
* Transizione o azione
* ID della funzione
* Tipo di elemento
* Contenuto rilevante con il termine di ricerca evidenziato

1. È possibile cliccare su qualsiasi risultato per visualizzare i dettagli completi in una finestra modale
2. Per esportare i risultati in formato testuale, cliccare sul pulsante "Export to Text" presente in fondo alla pagina

## File dei risultati

I risultati mostrano dove il termine di ricerca (es. "8080") appare nei workflow di Jira. Tipicamente questi riferimenti si trovano nelle Post-Functions.

**Risoluzione dei problemi**

* **Nessun file trovato**: Assicurarsi che i file XML siano stati correttamente estratti nella cartella xml
* **Errori di decodifica**: Alcuni contenuti Base64 potrebbero non essere decodificabili, ma lo script continuerà comunque l'elaborazione
* **Nessun risultato trovato**: Se non viene trovato alcun risultato, provare con varianti del termine di ricerca (es. "localhost:8080", "<http://server:8080>")

### Esempio del report “result.html” generato: Immagine che contiene testo, schermata, numero, software Descrizione generata automaticamente

### Come esportare il report in formato .txt tramite button Immagine che contiene testo, schermata, Carattere Descrizione generata automaticamente

Il file esportato avrà una nomenclatura come questa: “*search\_results\_8080\_1747321156209.txt*”.