Using Windows® PowerShell scripts in Fulfillment Workflow

Front Office

Biomni Ltd, 4 Bloomsbury Square, London, WC1A 2RP -T: +44(0) 20 7557 4200 F +44(0) 20 7557 4201 E: info@biomni.com

Contents

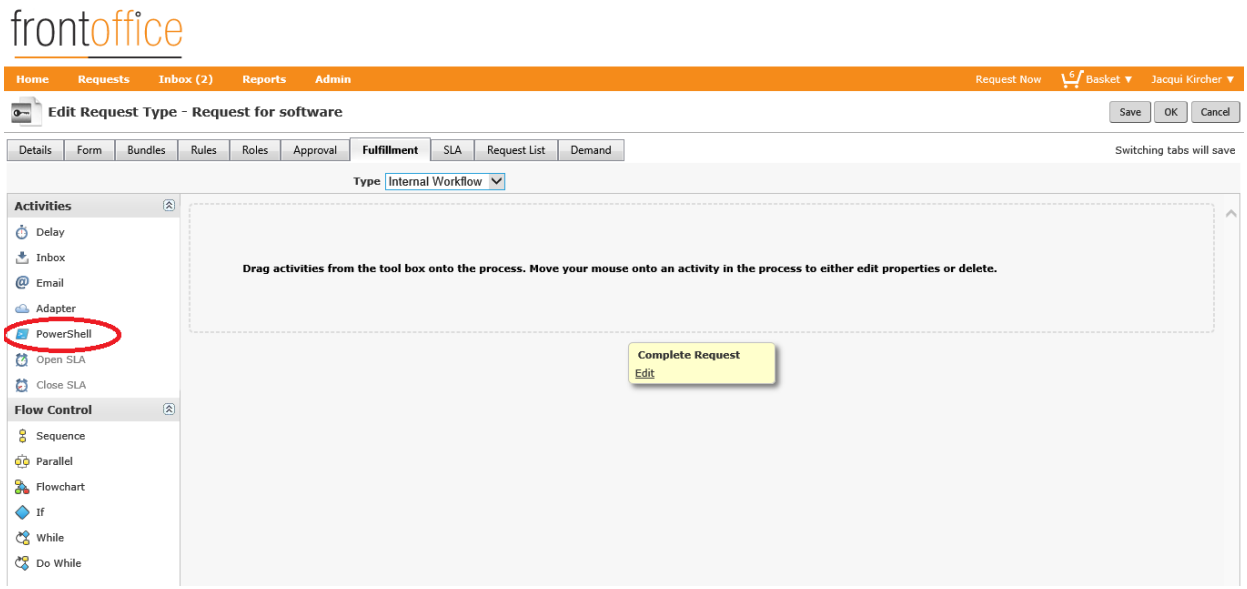
1.0 Introduction

Since version v7.3, the Front Office Fulfillment Workflow supports the use of PowerShell scripts as an activity. This document describes how to use PowerShell to interact with the Front Office API. A sample PowerShell script will be used throughout to demonstrate the various touch points between PowerShell and Front Office.

It is assumed that the server running Front Office has PowerShell already installed; this comes as standard with Windows 2008 systems.

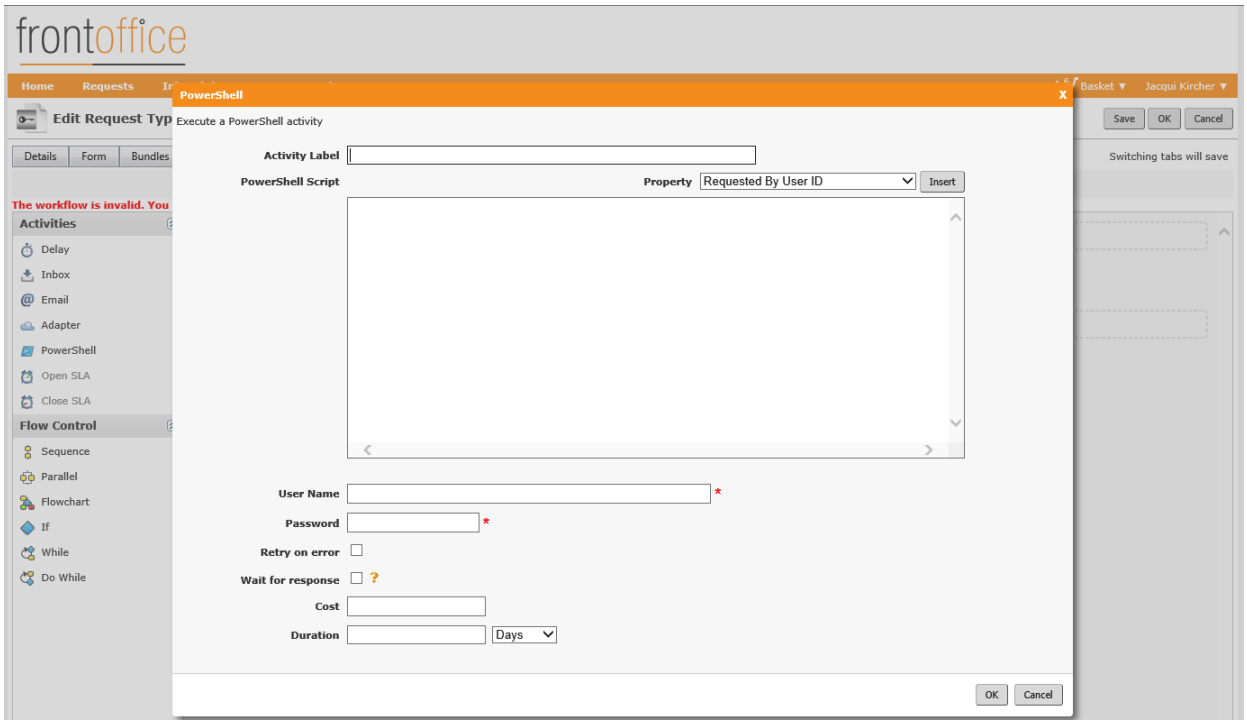
2.0 Creating a PowerShell activity

A PowerShell activity is available in the Fulfilment section when editing a Request Type.



Like any other activity, the user drags and adds it the workflow. In this example, only one activity will be used.

As soon as the PowerShell activity is dragged on to the workflow, the activity description page opens:



Unlike the Web Service activity which contains a reference to an externally hosted instance, in a PowerShell activity the script itself is stored within the activity description.

2.1 Activity Details

Activity Label: a textual description of the activity.

PowerShell Script: the PowerShell that will run when the activity is executed.

Property: provides access to various request and environment variables. They act as replacement parameters i.e. they are replaced by the corresponding request values at run time.

User Name & *Password*: the UserID and password for the user under whose account the PowerShell script will be executed.

Wait for response: checkbox to indicate the activity will wait for a complete signal from an external system. By default the activity completes as soon as the PowerShell script finishes.

Cost: an indication of the activity monetary value.

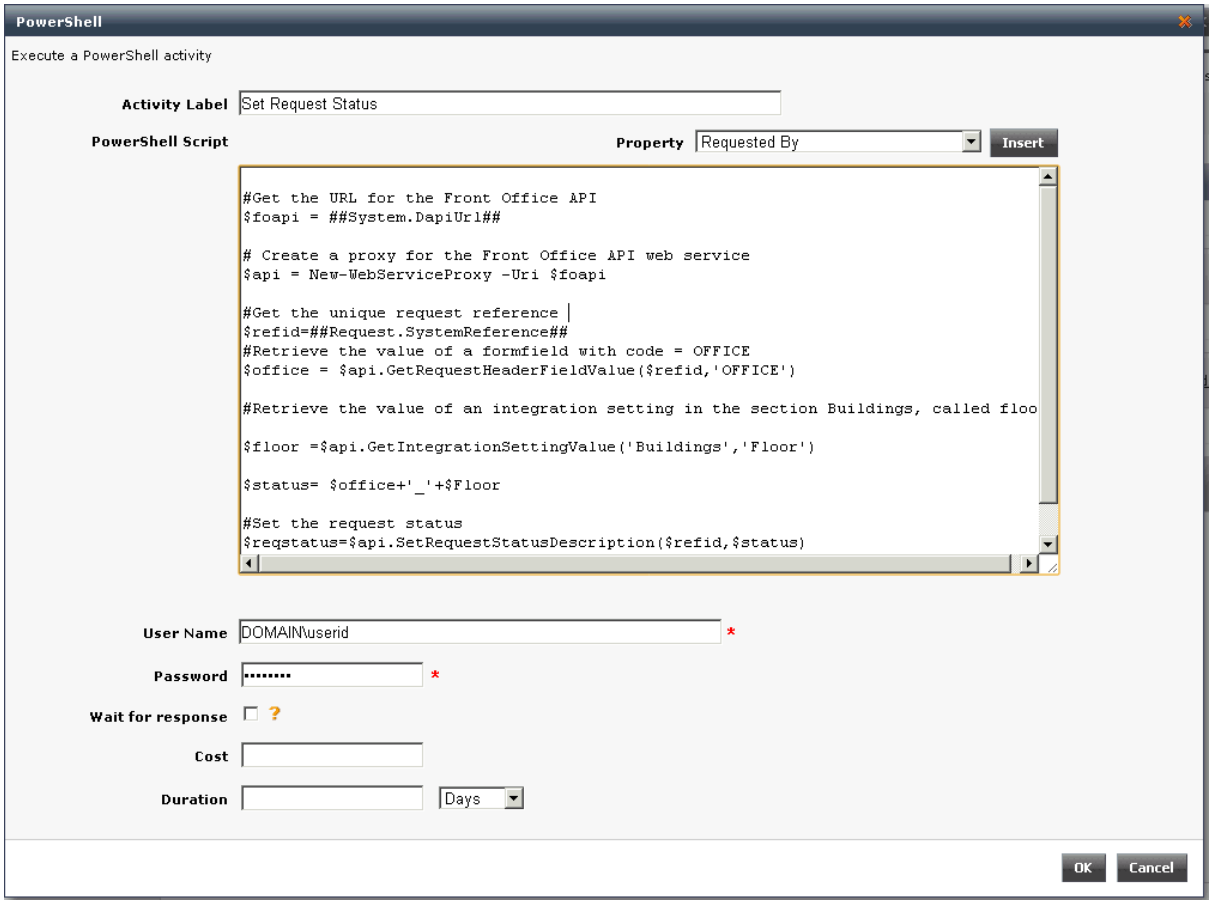
Duration: an indication of how long it takes to complete the activity.

2.2 Security

The PowerShell script will be executed on the application server that is hosting the Front Office instance. It is therefore important to run the script under an account which has the right to run scripts and also exclude from the account's access rights the areas of the system typically managed by IT e.g. formatting a disk or accessing IIS. It is the responsibility of the system designers to create an account with the appropriate rights.

3.0 Using the PowerShell activity

In this example, PowerShell is used to set the request status. This demonstrates how to use the Front Office API and replacement parameters.



Typically the script will be developed and tested outside Front Office and once verified, will be copy / pasted into the activity. Replacement parameters work only inside Front Office so it is recommended that variables are created and assigned to the replacement variables. Then when developing the script the replacement parameters can be replaced by hard coded values for testing purposes.

This is the script that is used in the above example:

```
#Get the URL for the Front Office API
$foapi = '##System.DapiUrl##'

# Create a proxy for the Front Office API web service
$api = New-WebServiceProxy -Uri $foapi

#Get the unique request reference
$refid=##Request.SystemReference##
#Retrieve the value of a formfield with code = OFFICE
$office = $api.GetRequestHeaderFieldValue($refid,'OFFICE')

#Retrieve the value of an integration setting in the section
Buildings, called floor

$floor =$api.GetIntegrationSettingValue('Buildings','Floor')

$status= $office+'_'+$Floor

#Set the request status
$reqstatus=$api.SetRequestStatusDescription($refid,$status)
```

A typical example for this approach is the unique request reference which is used in almost every API call that is related to the active request.

So assign the unique system reference to an internal variable:

```
$refid=##Request.SystemReference##
```

In the test environment a hard coded value is used for testing:

```
$refid='289034'
```

This means that when the script is pasted into the activity, only the hardcoded value needs to be replaced, as the internal value is used everywhere else in the script.

4.0 Calling the Front Office API

The Front Office (Directa) API is a web service and calling a web service from within a PowerShell script is straight forward:

1. Create a proxy for the web service
   ```
   $api = New-WebServiceProxy -Uri $foapi
   ```
2. Call any method
   ```
   $office = $api.GetRequestHeaderFieldValue($refid,'OFFICE')
   ```

Depending on the PowerShell IDE in use, Intellisense can make it easier to retrieve the methods available.

The Front Office SDK can be found in <installlocation>\SDK