File Access Manager

Access Fulfillment Using a Script

Version: 8.3 Revised: March 29, 2022

Access Fulfillment for Unmanaged Business Resources in File Access Manager

File Access Manager supports automatic access fulfillment for unmanaged business resources, using a user script.

The stages of approval remain as they are for managed business resources.

See the chapter on fulfillment in the File Access Manager Administrator Guide.

Configuration

Manual and automated fulfillment options on the access request template

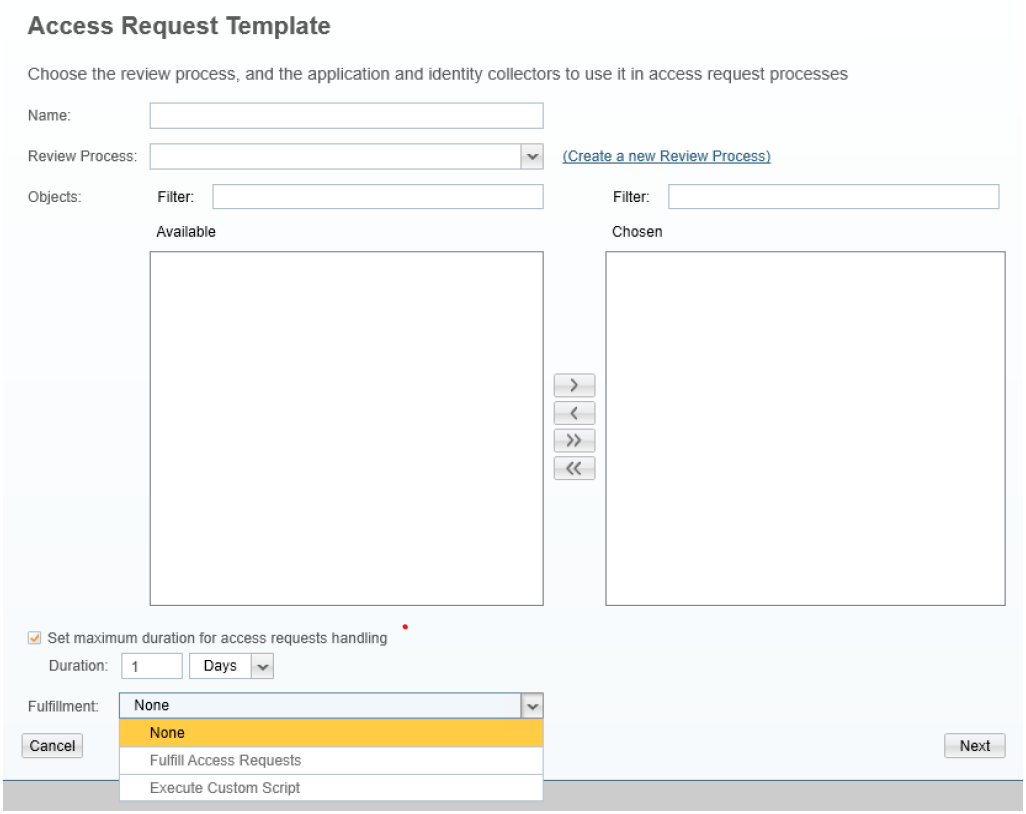
Fulfillment field	Managed BRs	Unmanaged BRs
None	No action	No action
Fulfill Access Request	Fulfillment processed automatically by the system	Manual fulfillment process. The user performing the fulfillment has to mark the task as done.
Execute Custom Script	Fulfillment processed automatically by the system	Fulfillment processed automatically, calling the custom script for each BR.

How to Set Up Fulfillment Using a Script

1. Open the Access Request Template

   Access Requests >> Configuration >> Manage Access Requests Templates

2. Double click an existing template to edit it, or click **New** to create a new template.
3. In the **Fulfillment** field, select **Execute Custom Script**.

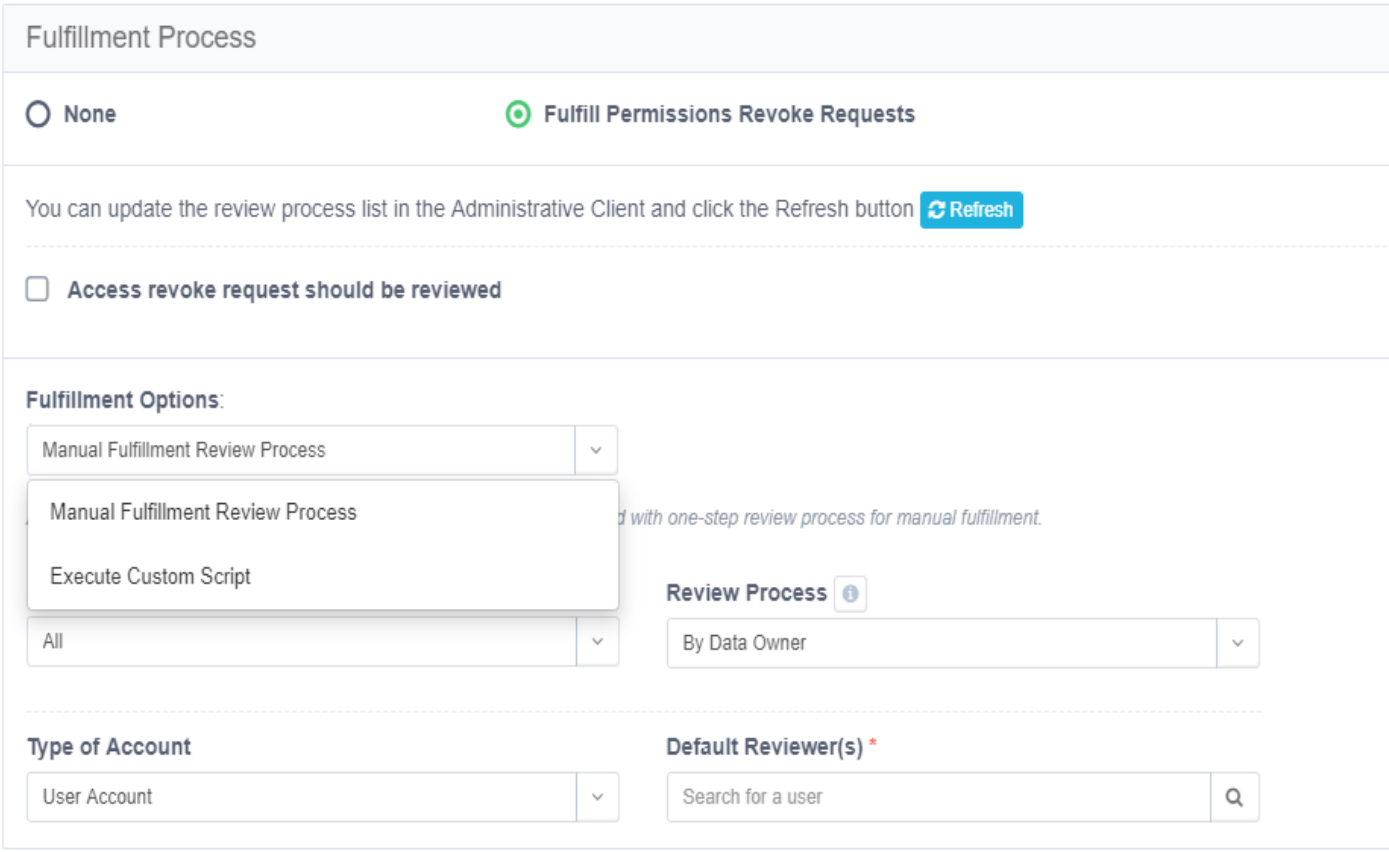


This will assume the script, named “Custom-Fulfillment.ps1” is in the required folder.

4. In the Access Certification campaign management:

   Compliance > Access Certification > Campaign Management > Manage Access Requests Templates

5. Edit an existing campaign, or click **+New Campaign**.
6. When you get to the **Summary** tab, open the **Fulfillment** option, by clicking **Edit**.
7. Select **Fulfill Permissions Revoke Requests**.
8. In the **Fulfillment Options**, select **Execute Custom Script**.



> For managed BRs, this campaign will automatically revoke the permissions from users selected

> Users from BRs that are not managed that were selected to revoke permission will be processed using this user script, as described above.

Script location

The user script has to be stored in the folder %SAILPOINT_HOME%\%SAILPOINT_APP_NAME% \Sched-uledTaskHandler

There is a sample script in that folder that comes with the installation package.

Script sample and input / output variables

```
<#
.SYNOPSIS
Custom-Fulfillment.ps1 - Changes a file or folder ACL.

.DESCRIPTION
This script will modify the security descriptor of a specified item, such as a file or a
folder, to match the values that have been supplied.

.INPUTS
This script gets a list of parameters as described below
.OUTPUTS
The script will return a string or integer value:
    Success = 0
    Error != 0

.NOTES
Written by: SailPoint Technologies
#>
# Main
 param (
    [bool]    $isRollback,        # Determines whether this is a rollback action or not
    [string]$actionType,        # The action type performed (AddPermission, RemovePer-
mission, AddUserToGroup, RemoveUserFromGroup)
    [string]$requestedBy,        # The user that created the access request
    [string]$applicationName,    # The request application name
    [string]$applicationType,    # The request application type (e.g. FILES MINI-FILTER)
    [string]$resourceFullPath,    # The full path of the resource (will be empty in case
the request is performed on a group)
    [string]$permissionType,    # The type of permission to add/remove
    [string]$accessRequestID,    # The access request ID
    [string]$campaignName,        # The campaign name (will be empty if the request
hasn't been created from a campaign)
    [string]$filterView,        # The campaign filter view type (FineGrained, User-
sAndRoles, Users, FineGrainedWithEveryone)
    # User fields in which the action is performed on (will be empty if the action is not
performed on a user)
    [string]$user,                # The user name
    [string]$userFullName,        # The user full name
    [string]$userUID,            # The user unique identifier
    [string]$userDisplayName,    # The user display name
    [string]$userPrincipalName, # The user principal name
    [string]$userType,            # The user entity type name
    [string]$userField1,        # The user enrichment fields (1 - 32)
    [string]$userField2,
  #  (This goes on for a while)
    [string]$userField31,
    [string]$userField32,
    # Group fields in which the action is performed on (will be empty if the action is
not performed on a group)
    [string]$group,            # The group name
    [string]$groupUID,        # The group unique identifier
    [string]$groupType,        # The group type
    [string]$groupDomain,    # The group domain
```

```
    [string]$groupField1,    # The group enrichment fields (1 - 10)
    [string]$groupField2,
    [string]$groupField3,
# and so on...
    [string]$groupField9,
    [string]$groupField10
)
#####################################
# Start writing your code from here #
#####################################
if (($actionType -eq 'AddPermission') -and ($user -ne '')) {
    if ($isRollback -eq $false) {
        # Adds permission to the specified user
    }
    else {
        # Handle rollback for the current action
    }
}
if ($success) {
    return 0; # Success
}
else {
    return 1; # Failure
}
```

Returned Values

The following codes are returned from the script:

0

Success

1-7

Error values