Ulf Lindström

Ulf.lindstrom@gmail.com

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Generic Powershell Extensions

Forefront Identity Manager 2010

Table of Contents

[Installation 3](#_Toc371634397)

[workflow activity 3](#_Toc371634398)

[*Register dll in GAC* 3](#_Toc371634399)

[*Register Activity Information Configuration* 3](#_Toc371634400)

[Management Agent Extensible Connectivity 3](#_Toc371634401)

[Create new management agent in FIM 3](#_Toc371634402)

[Powershell Activity 4](#_Toc371634403)

[Configuration 4](#_Toc371634404)

[Create new PowerShell Activity Workflow 4](#_Toc371634405)

[Powershell Scripts – Workflow Activity 7](#_Toc371634406)

[Parameter syntax 7](#_Toc371634407)

[Powershell – Extensible Management agent 8](#_Toc371634408)

[Configuration 8](#_Toc371634409)

[Create new powershell extensible management agent 8](#_Toc371634410)

[Powershell scripts – Extensible Management agent 10](#_Toc371634411)

[Schema generation powershell 10](#_Toc371634412)

[Export Powershell script 10](#_Toc371634413)

[Full Import Powershell Script 11](#_Toc371634414)

# Document description

This document draft describes the usage of Powershell extensions for FIM 2010 R2.

# Installation

## workflow activity

### Register dll in GAC

* Copy .\Images\\*.gif to "C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\TEMPLATE\IMAGES\Custom\"
* Create folder Custom under "C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\TEMPLATE\IMAGES\”
* Stop "Forefront Identity Manager Service" service
* ***Register dll into GAC***"C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\x64\gacutil.exe" /i {Path to dll} /f
* Start "Forefront Identity Manager Service" service
* ***Restart IIS***  
  iisreset

### Register Activity Information Configuration

* **Create new ”Activity Information Configuration” in FIM Portal**
* **Specify following settings**  
    
  DisplayName: Custom Powershell Activity {Or whatever you want to call it}  
  Activity name: FIMActivitiesLibrary.PowershellActivity  
  Assembly Name: FIMActivitiesLibrary, Version=1.0.0.0, Culture=neutral,  
   PublicKeyToken=65025db8603fc9b7  
  Type Name: WebUI.PowershellActivitySettingsPart
* **Restart IIS**  
  iisreset

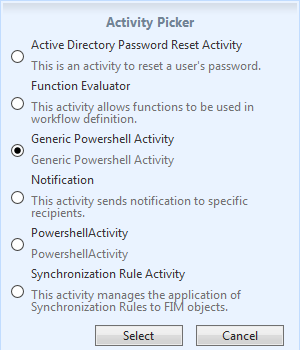
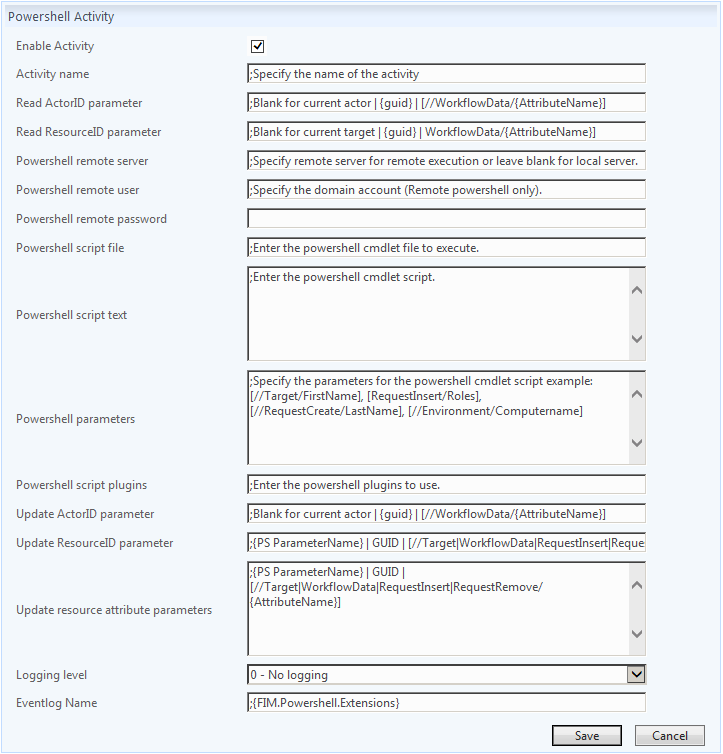
## Management Agent Extensible Connectivity

### Create new management agent in FIM

# Powershell Activity

## Configuration

### Create new PowerShell Activity Workflow

* Create new Workflow Activity and select the new activity  
  
* Edit the following parameters  
  

Parameters:

|  |  |
| --- | --- |
| Parameter | Usage |
| Enable Activity | Enables the execution of the PowerShell activity even if there is an MPR who triggers it. |
| Activity Name | Name of the PowerShell activity |
| Read ActorID Parameter | The FIM GUID to act as when performing read on target object  *\*If not specified the default administrator GUID is used*  *Usage:*   |  |  | | --- | --- | | *GUID Example : “63fb080c-f578-4398-8ef5-f1229b19488e”* | *Specifies an direct GUID to use* | | *[//WorkflowData/{AttributeName}]* | *Gets the GUID from the workflow data which had been set before this activity* | |
| Read ResourceID Parameter | Specifies the target the read operation should read from.  can specify the GUID of that target object you wish the read from.  *\*If not specified the current target in the workflow is used.*  *Usage:*   |  |  | | --- | --- | | *GUID Example : “63fb080c-f578-4398-8ef5-f1229b19488e”* | *Specifies an direct GUID to use* | | *[//WorkflowData/{AttributeName}]* | *Gets the GUID from the workflow data which had been set before this activity* | |
| Remote Server | The DNS or IP address to the server running the remote PowerShell service, requires the installation and configuration of WinRM on specified server. <http://msdn.microsoft.com/en-us/library/aa384372(v=vs.85).aspx> |
| Remote User | The account to run the PowerShell script on remote server |
| Remote Password | The password to use |
| Cmdlet file | The full path to the PowerShell script to be executed |
| Cmdlet script | The Powershell script to be executed |
| PowerShell Parameters | Specifies the parameters to use as input parameters on PowerShell script  *Usage: {Parameter} [//{Source}/{Attribute}]*   |  |  | | --- | --- | | Source | Description | | Request | Values from all types of requests  Example:  FirstName [//Request/FirstName] | | RequestModify | Values from modify on single-valued attribute Example:  LastName [//Request/LastName] | | RequestInsert | Contains inserted values on a multivalued attribute Example:  Roles [//RequestInsert/Roles] | | RequestRemove | Contains removed values on a multivalued attribute Example:  Roles [//RequestRemove/Roles] | | RequestCreate | Contains all attribute values from a create request  Example:  CreateAccountName [//RequestCreate/AccountName] | | Environment | Any windows environment variable Example :  ComputerName [//environment/computername] | | “Static Text” | Statisk text Example:  Country “Sweden” | | WorkflowData | Values from workflowdata Example:  ManagerEmail [//WorkflowData/ManagerEmail] | | Operation | Values from the operation   |  |  | | --- | --- | | Requestor | Guid of the object requesting the change | | Target | Guid of the target object | | TargetObjectType | Object type of target | | OperationType | Type of operation | | CreatedTime | Creation time of workflow | | RequestStatus | Status of the request | | | Function | Function values   |  |  | | --- | --- | | LogLevel | Configured logging level | | ActivityName | Name of the activity | | |
| Plugins | The plugins that will be added to the PowerShell and to be used by script |
| Update ActorID Parameter | The FIM GUID to act as when performing update on target object  *\*If not specified the default administrator GUID is used*  *Usage:*   |  |  | | --- | --- | | *GUID Example : “63fb080c-f578-4398-8ef5-f1229b19488e”* | *Specifies an direct GUID to use* | | *[//WorkflowData/{AttributeName}]* | *Gets the GUID from the workflow data which had been set before this activity* | |
| Update ResourceID Parameter | Specifies the target the operation should update.  *\*If not specified the current target in the workflow is used.*  *Usage:*   |  |  | | --- | --- | | *GUID Example : “63fb080c-f578-4398-8ef5-f1229b19488e”* | *Specifies an direct GUID to use* | | *[//WorkflowData/{AttributeName}]* | *Gets the GUID from the workflow data which had been set before this activity* | |
| Update resource attribute parameters | Specifies the parameters to use for update target resource  *Usage: {Parameter} [//{Source}/{Attribute}]*   |  |  | | --- | --- | | Source | Description | | Target | Set target attribute value  *Example:*  psFirstName [//Target/FirstName] | | RequestInsert | Inserted value(s) to a target multivalued attribute  *Example:*  psInsertRoles [//TargetInsert/Roles] | | RequestRemove | Removes value(s) from a target multivalued attribute  *Example:*  psRemoveRoles [//TargetRemove/Roles] | | WorkflowData | Set WorkflowData attribute value or replace current value if exists  *Example:*  psManagerEmail [//WorkflowData/ManagerEmail] | |
| Logging Level | 0-4 |
| Eventlog Name | Specifies in witch eventlog to create events  \*Default if not specified is Application log. |

## Powershell Scripts – Workflow Activity

### Parameter syntax

##### Example: Input parameter handling

PARAM($InsertValues,$RemoveValues,[string]$InsertValue,[string]$RemoveValue,$Department)

##### Example: building Return psObject hashmap

#Create the multi-value arrays

$stringArray = @("IT", "Helpdesk", "Support")

$intArray = @(1, 2, 3)

$stringValue = “Test”

$intValue = “Test”

#Create new array to return

$result = @{}

#Add values to multi-valued attribute

if($stringArray -ne $null) { $result.Add("stringValues",$ stringArray)

if($intArray -ne $null) { $result.Add("intValues",$ intArray)

#Add value to single-valued attribute

if($intValue -ne $null) { $result.Add("intValue",$intValue)

if($stringValue -ne $null) { $result.Add("stringValue",$stringValue)

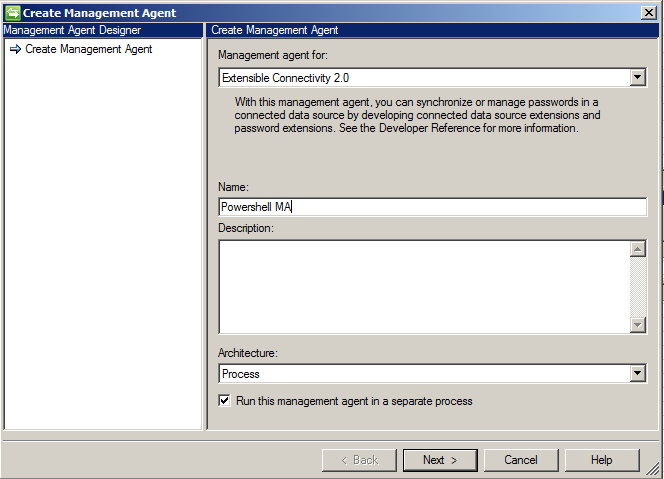
#Return values a hash-map to workflow activity

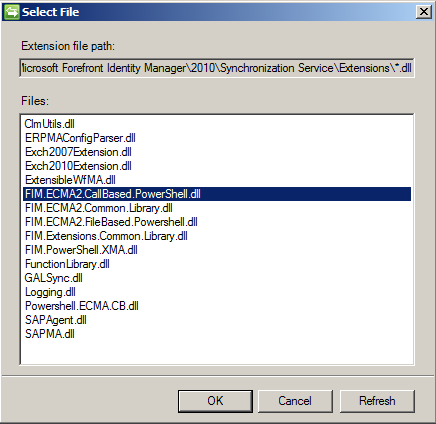
$result

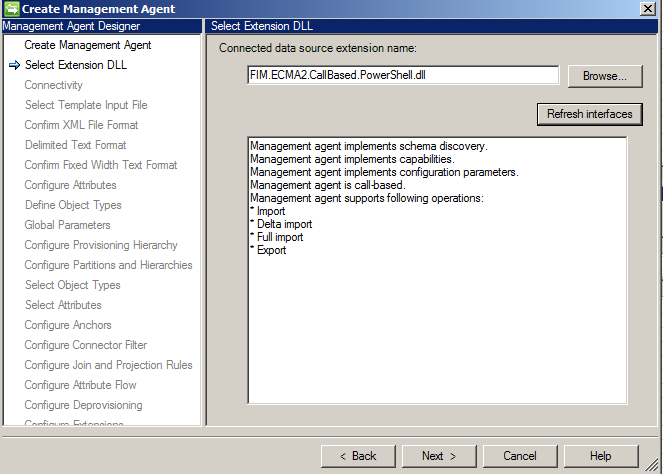
# Powershell – Extensible Management agent

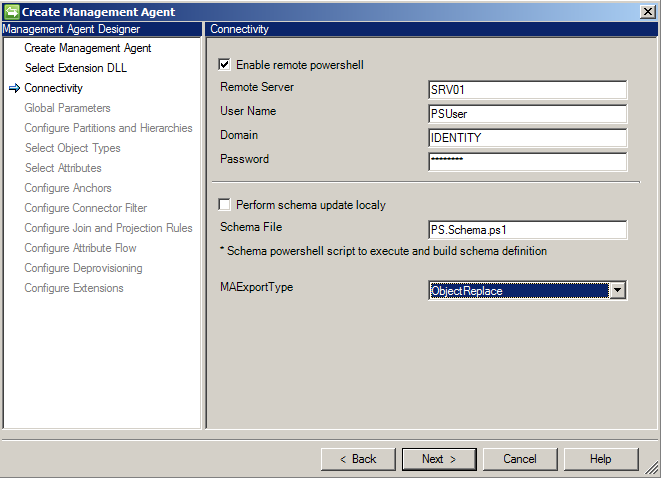
## Configuration

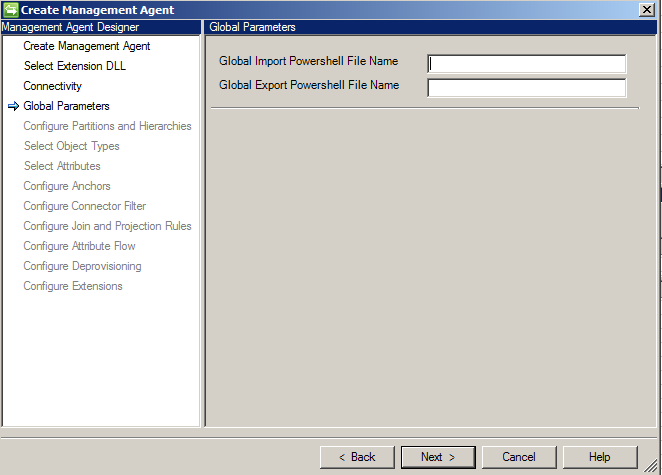
### Create new powershell extensible management agent

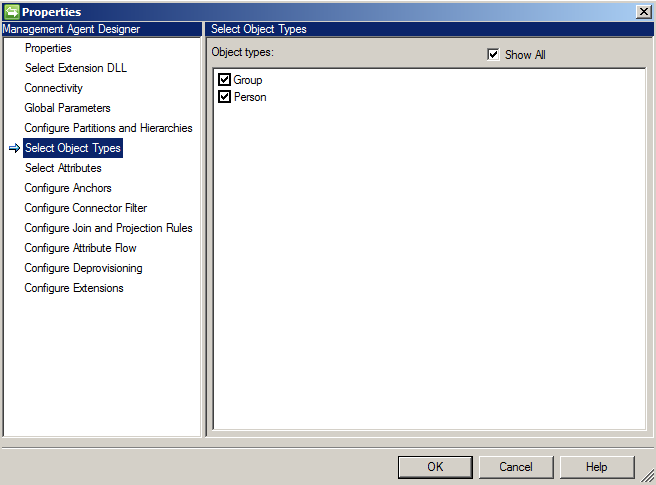


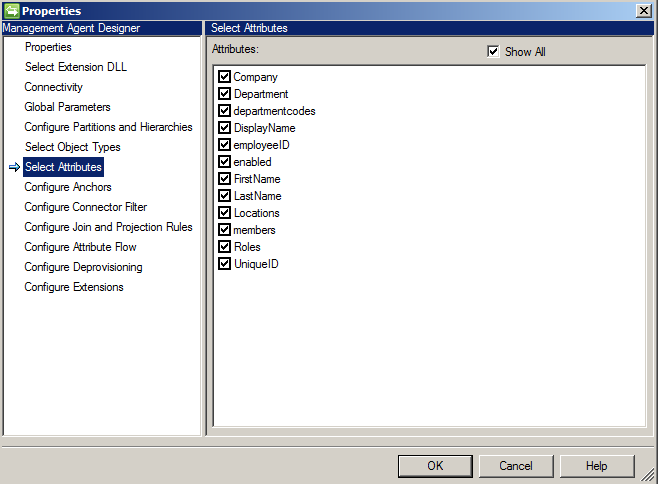


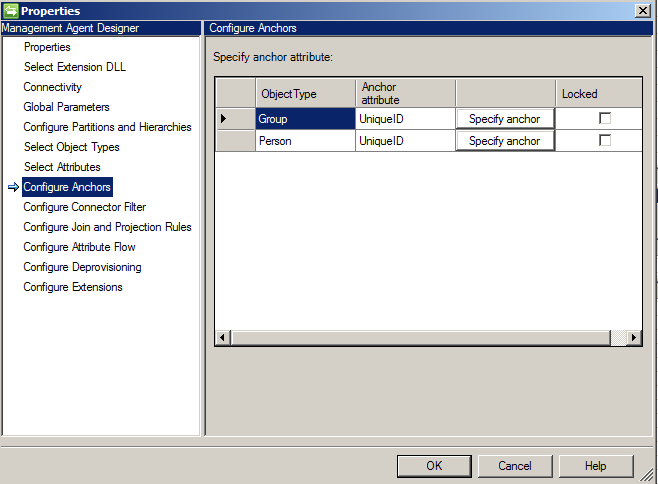












## Powershell scripts – Extensible Management agent

### Schema generation powershell

# Define attributes for Person object

$PersonAttributes = @(

"name=UniqueID,type=String,anchor=true,multivalue=false";

"name=FirstName,type=String,anchor=false,multivalue=false";

"name=LastName,type=String,anchor=false,multivalue=false";

"name=Company,type=String,anchor=false,multivalue=false";

"name=Department,type=String,anchor=false,multivalue=false";

"name=Locations,type=String,anchor=false,multivalue=true";

"name=Roles,type=String,anchor=false,multivalue=true";

"name=employeeID,type=integer,anchor=false,multivalued=false";

"name=enabled,type=boolean,anchor=false,multivalued=false";

"name=departmentcodes,type=integer,anchor=false,multivalued=true";

)

#Define person object

$Person = @{"objType" = "Person"; "attributes" = $PersonAttributes}

# Define attributes for Group object

$GroupAttributes = @(

"name=UniqueID,type=String,anchor=true,multivalue=false";

"name=DisplayName,type=String,anchor=false,multivalue=false";

"name=members,type=String,anchor=false,multivalue=true";

)

#Define group object

$Group = @{"objType" = "Group"; "attributes" = $GroupAttributes}

#Return objects

$Person

$Group

### Export Powershell script

PARAM($ModificationType,$DN,[HashTable] $AnchorAttributes,[hashTable] $ChangedAttributes,[hashTable] $AddValues,[hashTable] $RemoveValues)

"### RunStep = $RunStepType, RunPartition = $RunStepPartition ###" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

"DN = $DN" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

"Object ModificationType = $ModificationType" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

foreach($key in $AnchorAttributes.keys)

{

$value=$AnchorAttributes[$key]

"Anchor attribute : $key, Value= $value" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

}

foreach($key in $ChangedAttributes.keys)

{

$value=$ChangedAttributes[$key]

"Attribute Changed : $key, Change Type= $value" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

}

foreach($key in $AddValues.Keys)

{

$values=$AddValues[$key]

foreach($value in $values)

{

"Add value $value to $key" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

}

}

foreach($key in $RemoveValues.Keys)

{

$values=$RemoveValues[$key]

foreach($value in $values)

{

"Remove value $value from $key" | out-file -FilePath "c:\ChangedAttributes.txt" -Append

}

}

"" | out-file -FilePath "c:\ChangedAttributes.txt" –Append

### Full Import Powershell Script

PARAM([string]$RunStepType,[int]$PageSize,[int]$ImportLastCount)

#Input Parameters

#

#RunStepType = Delta,Full,FullObject

#PageSize = Maximum amount of objects to return each time

#ImportLastCount = Zero based number pointing to next object to import

#

$DepartmentCodes1 = @([int]1,[int]5,[int]9.5)

$DepartmentCodes2 = @([int]20,[int]70,[int]100)

$Roles = @("IT", "Helpdesk", "Support")

$MaxResult = 50

for ($i=$ImportLastCount; $i -le ($ImportLastCount+$PageSize-1) -and ($i -lt $MaxResult); $i++)

{

$csObject = @{}

$csObject.Add(".objectType","Person")

$csObject.Add(".ObjectModificationType","Add")

$csObject.Add("FirstName","PSTest$i")

$csObject.Add("LastName","User$i")

$csObject.Add("UniqueID","PSTest$i")

$csObject.Add("Roles",$Roles)

$csObject.Add("Department","Department$1")

$csObject.Add("employeeID",[int]$i)

$csObject.Add("enabled",$true)

$csObject.Add("departmentcodes",$DepartmentCodes1)

$csObject

}