Service Automation

Training day 2

TOOLS4EVER

Training content day 2

- Delegated Forms advanced
- Data Sources advanced
- PowerShell tasks
- HelloID API’s

Recap day 1

- Approval workflows
- Self service products
- Dynamic Form
- Static data source
- Delegated Form basics

Comparison

Self Service product

- Self Service request
  - employee, manager, owner(s)
- Approval workflow
  - single- or multistep
- Additional request data
  - dynamic form
- Product status and actions
  - Approved, returned, etc

Delegated Form

- Helpdesk delegation
  - helpdesk or servicedesk
- Instant actions
  - no approval
- Enriched data
  - dynamic form
- “fire and forget”
  - only task history

HelloID Tasks / actions

TOOLS4EVER

HelloID Tasks / actions

- HelloID Task catalog
  - Including PowerShell (mostly used)
- Self service product state actions
  - Multiple actions per state (no sequence or data share between actions)
  - Requested, approved, returned, …
- Delegated Form action
  - Single action

HelloID Tasks / actions trouble shooting

- Request history
- Task history (products and scheduled tasks)
  - Process log
  - Summary log
  - Input variables overview
- Delegated Form (using new SA Agent infrastructure)
  - Activity details
  - Process logging
  - Audit logging
  - Form submission details
- Local PowerShell tooling

Custom Powershell best practices

- Fit for purpose
- Don’t create “generic scripts”
  - Copy paste and adjust
- Basic error handling
  - Try { } Catch { }
- Execute PS script on local server to confirm output

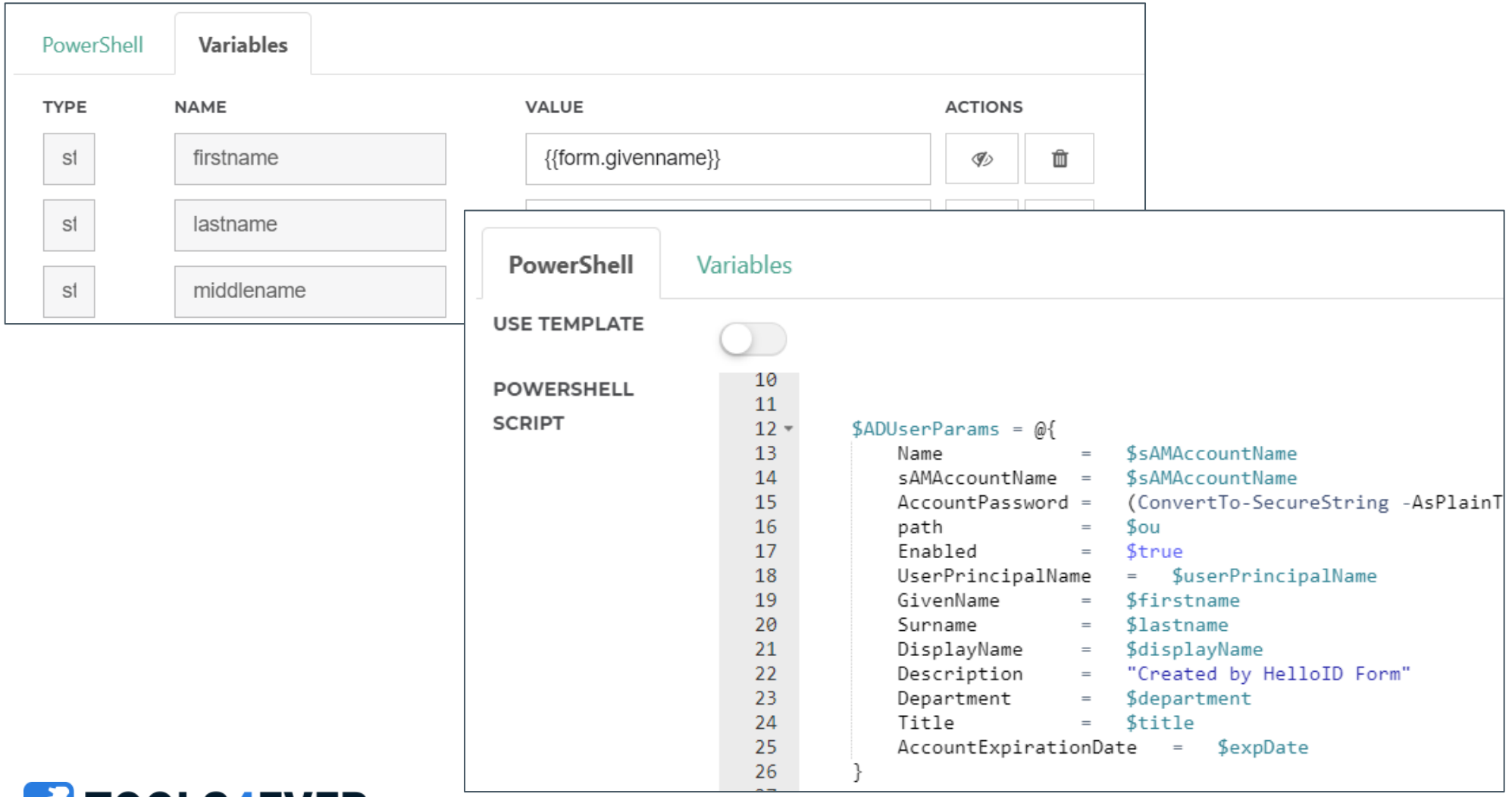
HelloID Task interaction

TOOLS4EVER

Using variables – Self Service products

- Global variables — no mapping needed. Example $ADuserUPNsuffix
- Form variables — example {{form.lastname}}
- Portal variables — example {{portal.baseUrl}}
- Requester — example {{requester.userName}}
- Manager — example {{manager.fullName}}
- Approval history — example {{request.approvalhistory.tojsonstring}}
- Product — example {{product.name}}
- Resource Owner Group — example {{resourceownergroup.groupGUID}}

Using variables



TOOLS4EVER

Products, scheduled tasks and task data source

- Write logs
  - Status log
  - Summary log
- Return data

```
Hid-Write-Status -Message "Result count: $resultCount" -Event Information
HID-Write-Summary -Message "Result count: $resultCount" -Event Information


Hid-Write-Status -Message "AD user [$sAMAccountName] created successfully" -Event Success
HID-Write-Summary -Message "AD user [$sAMAccountName] created successfully" -Event Success

HID-Write-Status -Message "Error searching AD users. Error: $($_.Exception.Message)" -Event Error
HID-Write-Summary -Message "Error searching AD users" -Event Failed



Hid-Add-TaskResult -ResultValue []
Hid-Add-TaskResult -ResultValue @{username = "demo01"; fullname = "Demo Account"; department = "Consultancy"}
```

Using variables – Delegated Forms

No mapping needed

• Global variables	$ADuserUPNsuffix
• Form variables	$form.lastname
• Portal variables	$portalBaseUrl
• Requester	$requester.userName

Delegated Forms and PowerShell data sources

- Write logs
  - Process log
  - Audit log
- Return data

```powershell
Write-Information "Result count: $resultCount"
Write-Warning "No results found"
Write-Error "Error searching AD users. Error: $($_.Exception.Message)"

$Log = @{
    Action = "CreateAccount" # optional. ENUM (undefined = default)
    System = "ActiveDirectory" # optional (free format text)
    Message = "Created account with username $userPrincipalName" # required (free format text)
    IsError = $false # optional. Elastic reporting purposes only. (default = $false. $true = Executed action returned an error)
    TargetDisplayName = $displayName # optional (free format text)
    TargetIdentifier = $createdSID # optional (free format text)
}
Write-Information -Tags "Audit" -MessageData $Log

Write-Output @{username = "demo01"; fullname = "Demo account"; department = "Consultancy"}
```

Lab 9

Complete Delegated Form “Create AD User”

TOOLS4EVER

Lab 9

45 minutes

Complete Delegated Form “Create AD User”

Description

- Create a (very) simple custom PS script to create an AD user account using the form inputs

Testing

- Log in as employee, open the Delegated Form and test the Form UI
- Check your Active Directory for new account

TOOLS4EVER

Please check out the references on https://docs.helloid.com
Manuals for Administrators → Automation → How to Use Variables
Manuals for Administrators → Automation → HelloID PowerShell Cmdlet Reference

Lab 9
Example

```powershell
# variables configured in form
$blnexpdate = $form.blnexpdate
$department = $form.department
$displayName = $form.naming.displayname
$expiredate = $form.expiredate
$firstname = $form.givenname
$lastname = $form.lastname
$middlename = $form.middlename
$ou = $form.ou.Path
$password = $form.password
$sAMAccountName = $form.naming.samaccountname
$userPrincipalName = $form.naming.UserPrincipalName

try {
    if($blnexpdate -ne 'true'){
        $expDate = $null
    } else {
        $expDate = [datetime]$expiredate
    }

    Write-Information "Expiredate: $expDate"

    $ADUserParams = @{
        Name              =   $sAMAccountName
        sAMAccountName    =   $sAMAccountName
        AccountPassword   =   (ConvertTo-SecureString -AsPlainText $password -Force)
        path              =   $ou
        Enabled           =   $true
        UserPrincipalName =   $userPrincipalName
        GivenName         =   $firstname
        Surname           =   $lastname
        DisplayName       =   $displayName
        Description       =   "Created by HelloID Form"
        Department        =   $department
        Title             =   $title
        AccountExpirationDate   =   $expDate
    }

    $tmp = New-ADUser @ADUserParams
    Write-Information "AD user [$sAMAccountName] created successfully"
} catch {
    Write-Error "Error creating AD user [$sAMAccountName]. Error: $($_.Exception.Message)"
}
```

TOOLS4EVER

Delegated Form

Import external template

TOOLS4EVER

Import external Delegated Form template

- Easy import of Delegated Form templates
- Using HelloID API (API key required)
- All-in-one PowerShell script importing all required resources

Lab 10

Import external template “Active Directory - Create user”

TOOLS4EVER

Lab 10

30 minutes

Import external template “Active Directory - Create user”

Description

- Make sure you have generated an API key
- Copy the Post-setup configuration steps
- Run the All-in-one PowerShell script
- Configure the Post-setup configuration steps

Testing

- Confirm no error messages after executing all-in-one PowerShell script
- Login as Servicedesk. Open and submit the Delegated Form
- Confirm the new account in your Active Directory

TOOLS4EVER

Please check out the references on https://docs.helloid.com
Manuals for Administrators → Automation → How to Use Variables
Manuals for Administrators → Automation → HelloID PowerShell Cmdlet Reference

HelloID Data Sources

TOOLS4EVER

Data sources

- Static data source
- Task data source (old infrastructure)
- PowerShell data source

Static Data Source

- Static data (stored within HelloID) defined as JSON
- No input parameters are available
- Required model definition

Task Data Source (old infrastructure)

- Dynamic external data
- Using a seperate HelloID PowerShell task
- Executed by the Directory Agent
- Optional input parameters available
  - Form variables
  - Global variables (no mapping needed)
- Required model definition
- Using HelloID functions to receive data
  - Hid-Add-TaskResult
- Using HelloID functions to return log messages
  - Hid-Write-Status
  - Hid-Write-Summary

Powershell data source

- Dynamic external data
- Received by executing a PowerShell script (no HelloID task needed)
- Executed by the Service Automation Agent (websockets vs polling)
- Optional input parameters available
- Required model definition
- Using native PowerShell functions to receive data
  - Write-Output
- Using native PowerShell to return log messages
  - Write-Information (Warning/Error)

No influence on delay in external systems

No Caching available

Powershell data source

Variables

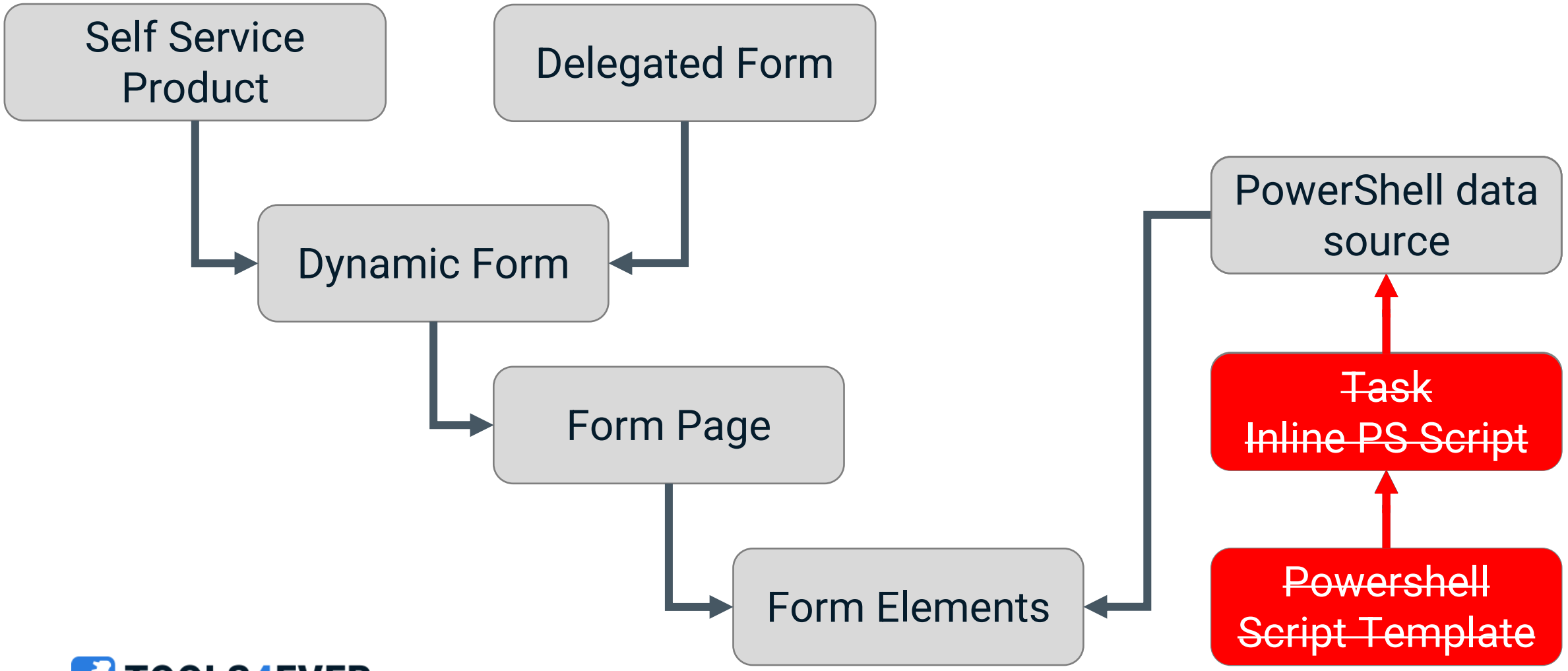
- Globale, user defined, variables → available
- $formInput. (for data source input) → $dataSource.
- Requester (without mapping) → $requester.

```powershell
Write-Information $requester

try {
    $searchValue = $dataSource.searchUser
    $searchQuery = "*$searchValue*"


    $users = Get-ADUser -Filter {Name -like $searchQuery -or DisplayName -like $searchQuery} -properties displayName, UserPrincipalName, Department, Title
    $users = $users | Sort-Object -Property DisplayName
    $resultCount = @($users).Count
    Write-Information "Result count: $resultCount"

    if($resultCount -gt 0){
        foreach($user in $users){
            $returnObject = @{displayName=$user.displayName; UserPrincipalName=$user.UserPrincipalName; Department=$user.Department; Title=$user.Title;}
            Write-Output $returnObject
        }
    }
} catch {
    $msg = "Error searching AD user [$searchValue]. Error: $($_.Exception.Message)"
    Write-Error $msg
}
```

Data source comparison

	Static data source	Task data source	PowerShell data source
Dynamic (external) data	No	Yes	Yes
Model definition	Yes	Yes	Yes
Input parameters	No	Yes	Yes
Agent service support	None	Directory Agent	SA Agent
Support PowerShell native communication	No	No	Yes

HelloID Service Automation Topology



TOOLS4EVER

Lab 11

Create a PowerShell data source for searching AD users

TOOLS4EVER

Lab 11

30 minutes

Create a PowerShell Data Source for searching AD users

Description

Create a new Delegated that shows all AD users from specific OU that matches the search value

- Create a new Delegated Form containing a text input and a grid form element
- Add a PowerShell data source to the grid
  - Model definition
    - displayName
    - userPrincipalName
    - department
    - title
  - Input variable “searchValue” linked to form text input

Testing

- Use the Data Source “Execute Task” button to test the script.
- Confirm Dynamic Form shows the returned data

Keep it simple

1. Return all users of AD
2. Return all users of specific OU
3. Return all users of specific OU matching search criteria

TOOLS4EVER

Please check out the references on https://docs.helloid.com
Manuals for Administrators → Self Service → Data Sources - Overview

Lab 11
Example (very simple)

```
Write-Information $requester

try {
    $searchValue = $dataSource.searchUser
    $searchQuery = "*$searchValue*"

    $users = Get-ADUser -Filter {Name -like $searchQuery -or DisplayName -like $searchQuery} -properties displayName, UserPrincipalName, Department, Title
    $users = $users | Sort-Object -Property DisplayName

    if($resultCount -gt 0){
        foreach($user in $users){
            $returnObject = @{displayName=$user.displayName; UserPrincipalName=$user.UserPrincipalName; Department=$user.Department; Title=$user.Title;}
            Write-Output $returnObject
        }
    }
} catch {
    $msg = "Error searching AD user [$searchValue]. Error: $($_.Exception.Message)"
    Write-Error $msg
}
```

Lab 12

Create a new delegated form for setting manager in AD

TOOLS4EVER

Lab 12

60 minutes

Create a new delegated form for setting manager in AD

Description

- Create a new Delegated Form
- Configure a new multi-step Dynamic Form form where you can
  - Find and select an AD user account
  - Select the new AD manager account
- Copy the previous data source script to find the AD user account
- Create a new data source to select the new manager account
- Create a (very) basic PS script to set the AD user's manager attribute

Testing

- Enter details in Delegated Form
- Submit Delegated From
- Check your AD

Keep it simple

1. Form UI
2. Data source find AD user
3. Data source select new manager
4. PS script update AD user's manager

TOOLS4EVER

Please check out the references on https://docs.helloid.com
Manuals for Administrators → Automation → How to Use Variables
Manuals for Administrators → Automation → HelloID PowerShell Cmdlet Reference

Lab 13

Create a new delegated form managing AD group memberships of an AD user account

TOOLS4EVER

Lab 13

60 minutes

Create a new delegated form managing AD user groupmemberships

Description

- Create a new delegated form where you can
  - Find and select an AD User
  - Show AD groups to add
  - Show AD groups who are already memberof
  - Perform membership mutations via Powershell
- Create the multi-step Dynamic Form
- Copy data source script to find the AD User
- Create a new data source to show all available AD Groups
- Create a new data source to show current AD user Group memberships
- Create a (very) basic PS script to update the AD user group memberships

Testing

- Enter details in Delegated Form
- Submit Delegated From
- Check your AD

Keep it simple

1. Form UI
2. Data sources
3. PS script updating user's groupmemberships

TOOLS4EVER

Please check out the references on https://docs.helloid.com
Manuals for Administrators → Automation → How to Use Variables
Manuals for Administrators → Automation → HelloID PowerShell Cmdlet Reference

Lab 14

API time

TOOLS4EVER

Lab 14

API time

?? minutes

Multiple HelloID API's

- Users
- Groups
- Products
- Tasks
- Forms
- Etc…

All-in one Powershell scripts using HelloID API's to create complete Delegated Forms

- https://docs.helloid.com/ Manuals for Administrators → Catalog of PS Scripts to Create Delegated Forms
- https://github.com/Tools4everBV

TOOLS4EVER

How to implement Service Automation

- Client needs?
  - Self Service
  - Delegated Forms
- Target systems
- Responsibilities
- Setup Identity Provider for HelloID

Unique selling points

- Self service **automation**
- Part of rich IDM HelloID platform
- Managed users and products insights
- Extensive workflow options
- API usage

Quick reference guide

- https://docs.helloid.com/
  - Manuals
  - Changelog
  - API docs
- https://feedback.helloid.com/
  - Feature request
- https://forum.helloid.com/
  - Technical Q&A Forum
- https://roadmap.helloid.com/
  - Roadmap overview
- https://github.com/Tools4everBV
  - Connector / Forms repositories
- https://helloid.statuspage.io/
- https://docs.helloid.com/hc/en-us/categories/360002805319-Training
  - HelloID training materials

TOOLS4EVER