



HelloID Training

Service Automation – March 2023

Agenda

- Introduction
- Day 1
 - Setup HelloID portal + agent
 - Self Service products
 - Dynamic Forms
 - Data sources basics
 - Delegated Forms basics
- Day 2
 - Delegated Forms advanced
 - Data Sources advanced
 - Powershell tasks
 - HelloID API's

Introduction

Tools4ever

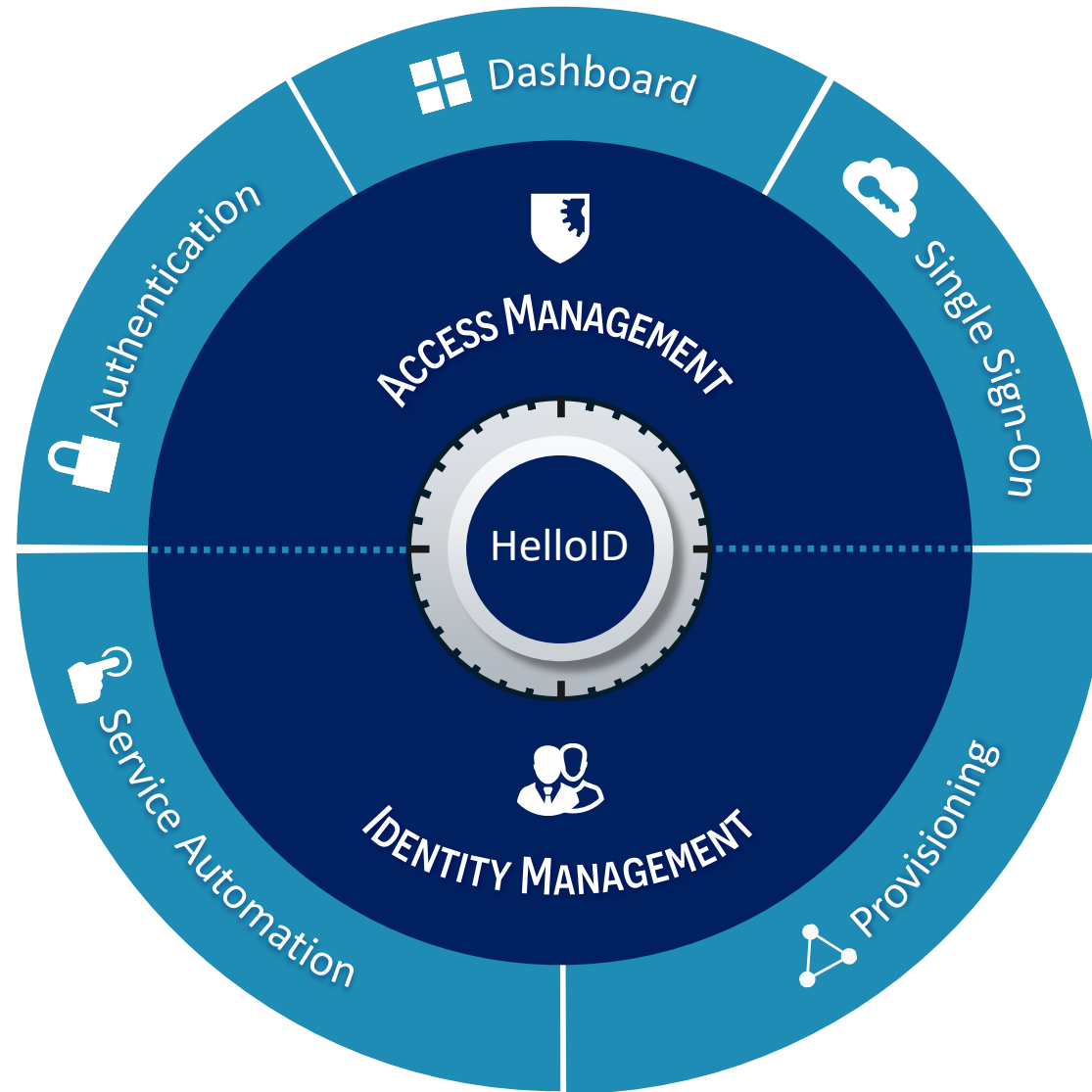
- Dutch origin
- Identity Management
- 7 sites worldwide
- 700 customers in NL
- 5000 customers worldwide
- 140 employees

HelloID Trainers

- Michiel van der Veecken

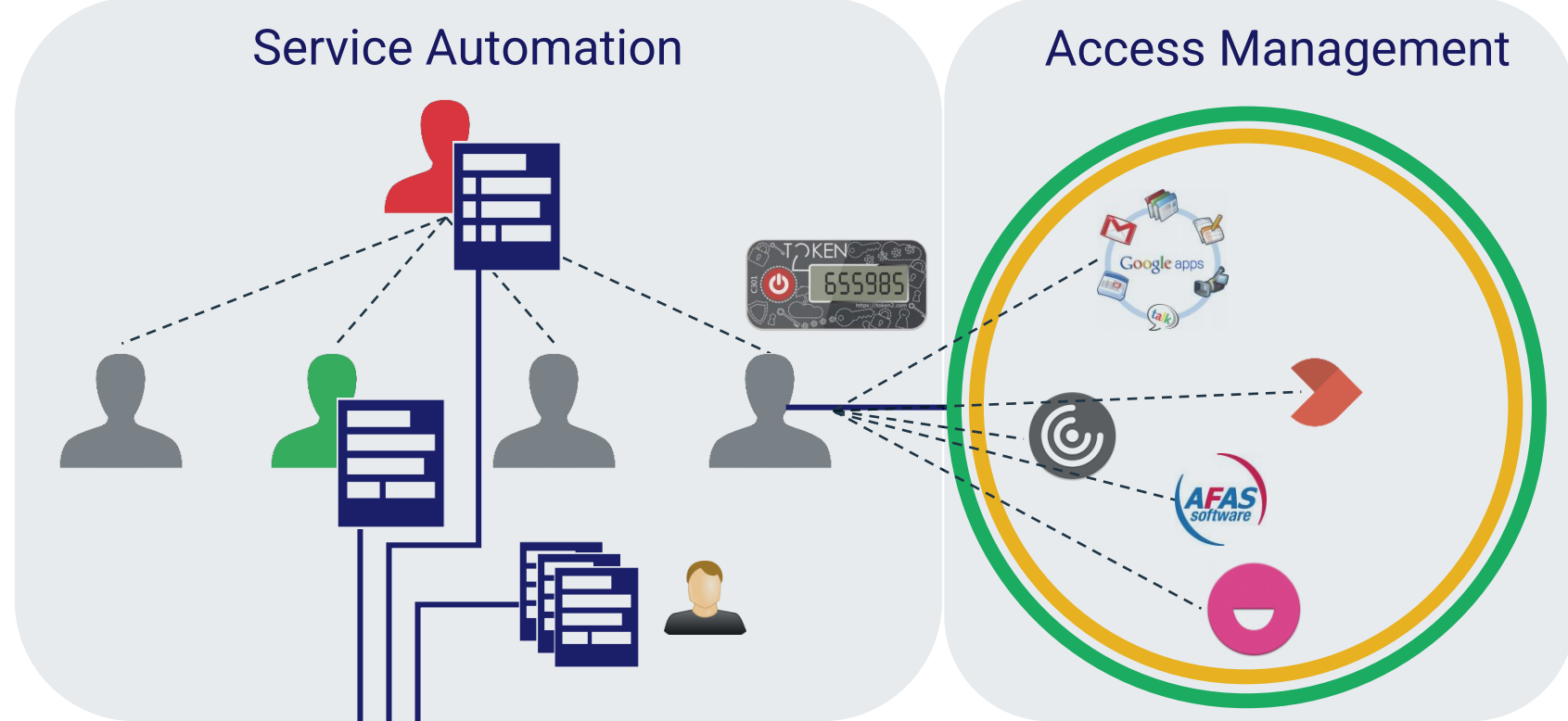
Audience

- Tools4ever partners
- Tools4ever customers

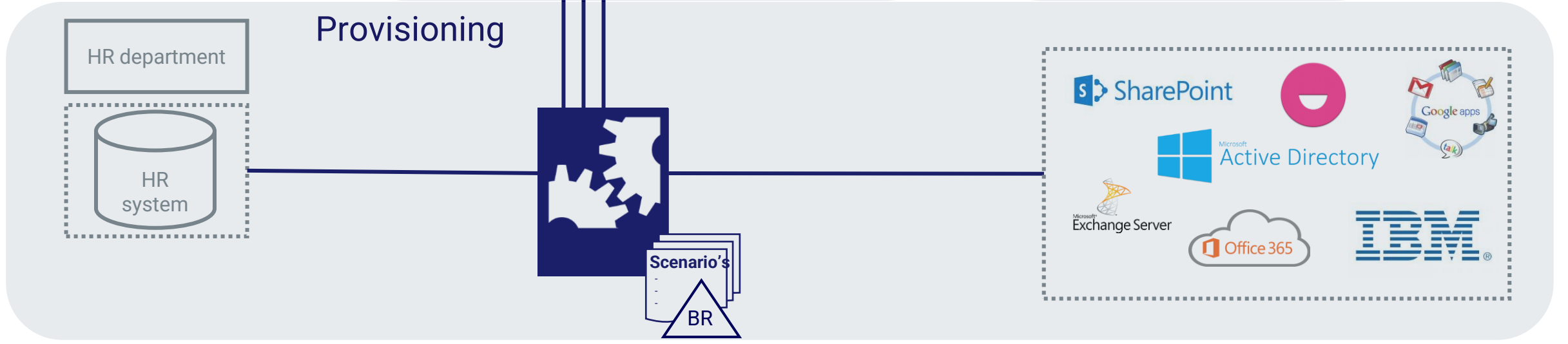


Service Automation

Access Management

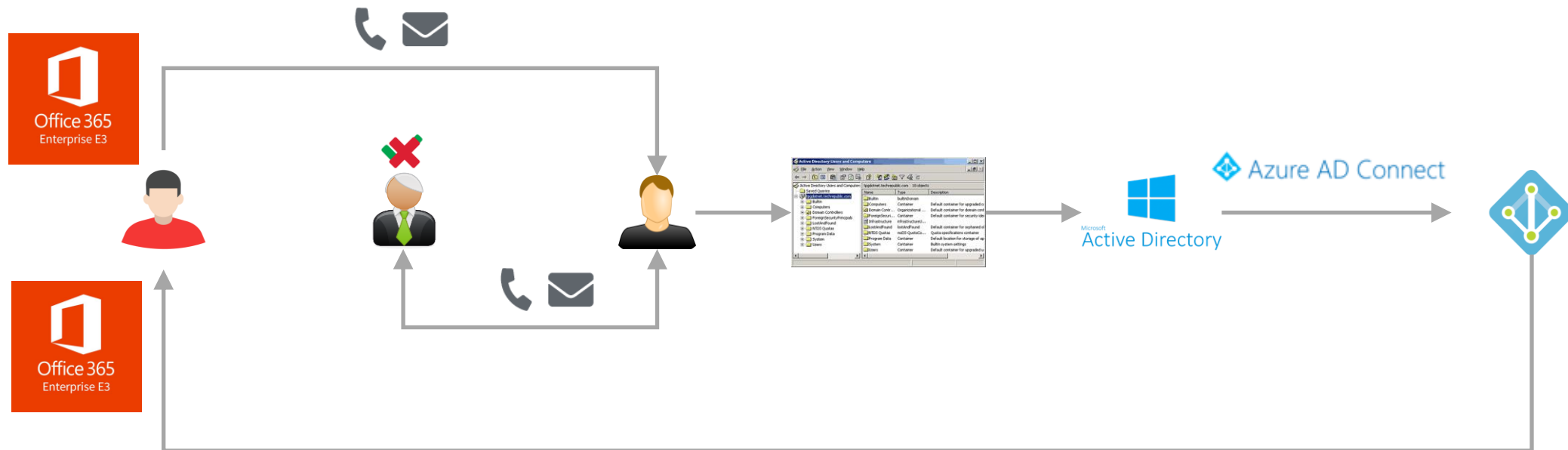


Provisioning



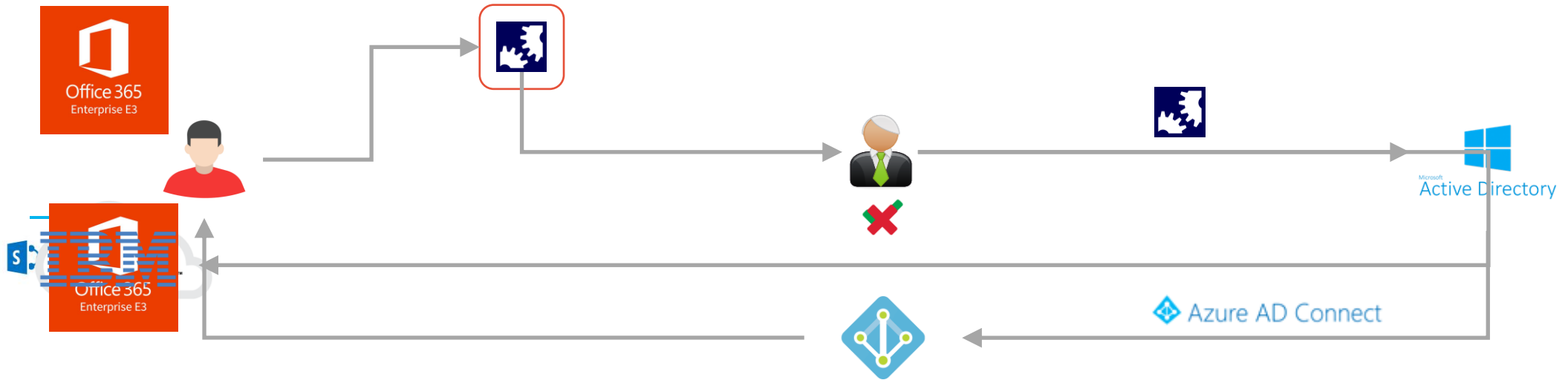
Service Automation

Without Service Automation





With Service Automation



Service Automation “modules”

- Self Service requests
- Helpdesk delegation
- Scheduled automation tasks
- (Scheduled) Reports

Service Automation

Training day 1

Training content day 1

- Setup HelloID portal + agent
- Self Service products
- Dynamic Forms
- Data sources
- Delegated Forms basics

How to setup HelloID

- HelloID Portal
- HelloID Agent
 - Agent Pool
 - Directory Configuration
- HelloID Roles
 - Employee
 - Manager
 - Servicedesk
 - Administrator
 - Consultant

HelloID Agent

HelloID Agent

- Directory Agent
- Provisioning Agent (on-prem and cloud)
- Service Automation agent (on-prem and cloud)

Directory Agent

- On-premise software
- Used for
 - AD authentication
 - AD user and group synchronization including mapping
 - Task execution (Scheduled and Products)
- Polling mechanism for task execution

Service Automation Agent

- On-premise **and cloud** software
- Used for
 - PowerShell data sources
 - Delegated Form task
- Websockets for faster response

Lab 1

Installing and configuring the HelloID Agent

Before we start... make sure your Active Directory contains the following (or use the provided PowerShell script)

OU structure

- HelloID Training
 - Users
 - Disabled users
 - Groups

AD user accounts

- 4 employee accounts
 - Configure department and title attributes
- 2 managers accounts
 - Configure department and title attributes
 - Configure manager attribute for user

AD Groups

- 4 AD groups for demo purpose
- Add groups to AD users (random)

Lab 1

30 minutes

Installing and configuring the HelloID Agent

Installing and configuring Active Directory Agent including sync of users and groups.

- Install the Active Directory Agent
- Create Active Directory configuration
 - Synchronize all user accounts from the OU “HelloID Training”
 - Synchronize all groups (because of group nesting)
 - Activate Authentication

Lab 2

Configure HelloID roles

Lab 2

Configure HelloID roles

15 minutes

Description

- Assign default Self Service role to all employees group
- Assign default Manager role to managers group
- Create “Servicedesk” role
 - Enable “Servicedesk → overview” right
 - Assign role to Servicedesk group
- Create “Super Admin” role
 - Enable all available rights
 - Assign role to administrator user

Testing

- Log in as employee. Do you see:
 - Applications
 - Service Automation → Self Service Products
- Log in as manager. Do you see:
 - Applications
 - Service Automation → Self Service Products
 - Management → Managed users
 - Management → Managed products
- Log in as servicedesk. Do you see:
 - Servicedesk?

Self Service products

Requestable "items" of any kind

End-user perspective

- Employee
 - Request / return product
 - View product request history
- Manager
 - Request / return product for managed employees
 - Inbox for approval actions
 - Approval actions are delegable
- Product owner
 - Request / return products owned by yourself
 - Inbox for approval actions
 - Approval actions are delegable

Admin perspective

HelloID admin portal

- Setup and configure HelloID
- Setup and configure Products, Workflows, Forms, Tasks, Scripts, etc...
- Request history
- Request Administration
- Override approval

How to setup a self service product

- Categories
- Approval Workflow
- Product scoping
- Product actions
- Dynamic Form (optional)

Lab 3

Creating an “instant approved” self service product

Lab 3

20 minutes

Creating an “instant approved” self service product

Description

- Create a new instant approved workflow
- Create a new category
- Create a new product and configure
 - Instant approved workflow
 - Icon
 - Access group for scoping
 - Allow multiple requests

Testing

- Log in as employee and request the product
 - Check your “My products” page
- Log in as manager and request the product for a the same user
 - Try again for a different user
- Log in as administrator and check
 - Self service administration
 - Request history

- No need to add actions / taks to the product
- No need to add a Dynamic form to the product

Lab 4

Creating a “multi-step approval” self service product

Lab 4a

20 minutes

Creating a “multi-step approval” self service product

Description

- Create a new multi-step approval workflow with the following approvers sequence
 - Requester’s manager
 - Resource owner
- Create a new product and configure
 - Multi-step approval workflow
 - Resource owner group
 - Icon
 - Access group for scoping
 - Allow multiple requests

Testing

- Log in as employee and request the product
 - Log in as manager and approve request
 - Log in as one of the resource owners and approve the request
- Log in as administrator and check
 - Self service administration
 - Request history

- No need to add actions / taks to the product
- No need to add a Dynamic form to the product

Lab 4b

15 minutes

Approval by email

Description

- Add your personal email address to an existing AD user account
- Run Active Directory synchronization
- Enable email notifications for your workflow

Testing

- Make a product request
- Approve or deny the request directly by email

- No need to add actions / tasks to the product
- No need to add a Dynamic form to the product

Dynamic Forms

Dynamic Forms basics

- Flexible form structure
- Single or multi page setup
- More than 18 form elements
- Row element (side by side)
- Data sources for enriched data
- Dynamic summary page

MEMBERSHIPS

Available	Member of
filter	filter
HelloID	GG_HelloID_Servicedesk
helloid_1	GG_Test
helloid_2	helloid_3
helloid_4	
jajajajajaj	
Janjansen	
Nogeenmooie.ADgrosSD	
proj_HelloID_SA	
test_dl	
test_dl_1	

Step 1: Details

ACCOUNT TYPE *

GIVENNAME *

John

MIDDLE NAME

van der

LAST NAME *

Poel

JOB TITLE

Application owner

DEPARTMENT

ICT

ACCOUNT EXPIRES

YES

PASSWORD *

Next →

Lab 5

Create a new dynamic form "Create AD user"

Lab 5

30 minutes

Create a new Dynamic Form “Create AD user”

Description

Create a new Dynamic form containing

- Text input for
 - Displayname, givenname and lastname
 - Username
 - Department and jobtitle
- Dropdown for AD location new account
 - Use static values within this form element, no data source (“live data”)
- Expire date configuration
 - Switch/toggle to enable expire date configuration
 - Date input connected to switch for conditional visibility
- Password field

- Graphical interface only
- No data source, actions or tasks required... yet

Lab 5

Example

DISPLAYNAME

USERNAME

DEPARTMENT

JOBTITLE

AD LOCATION *

EXPIREDATE?

CHECKBOX LABEL

EXIPREDATE

Data Sources

Data Sources

- Enriched data for Dynamic Forms
- Three types
 - Static data source
 - Task data source (old infrastructure)
 - PowerShell data source
- Input variables (task- and PowerShell data source)
- Model definition

Lab 6

Create a static data source

Lab 6a

15 minutes

Create a static data source

Description

- Open your Dynamic Form and edit the “AD OU location selector” Form element
- Alter the configuration and add a static data source
 - Containing attributes “path” and “name”
 - Path = distinguished name of AD OU
 - Name = friendly name to display in form
- Update static data source with data for “Enabled Users” and “Disabled User” information from your Active Directory

Testing

- Open (editor modus) the Dynamic Form and check the updated form element contains the static data source data

Lab 6a

Example

VALUE DEFINITION	MODEL DEFINITION			
<pre>1 [2 { 3 "path": "OU=Enabled users,OU=HelloID,DC=enyoi-media,DC=local", 4 "name": "Enabled" 5 }, 6 { 7 "path": "OU=Disabled users,OU=HelloID,DC=enyoi-media,DC=local", 8 "name": "Disabled" 9 } 10]</pre>	<table border="1"><tr><td>path</td></tr><tr><td>name</td></tr><tr><td>Add Field</td></tr></table>	path	name	Add Field
path				
name				
Add Field				

Lab 6b

15 minutes

Use data source default selection

Description

- Add a new column to the existing data source named “selected”
- Enter the numeric value 0 or 1 in your static data source value definition
 - Use value 1 for the default selected row
 - All other rows have value 0
 - **No quotes** around the 1 or 0
- Configure default selection in corresponding Form element

Testing

- Open (editor modus) the Dynamic Form and check the connected form element and it's default value

Delegated Forms

Delegated Forms

- Helpdesk delegation
- Categories
- No approval workflows
- Enriched data via dynamic forms
- Stateless “fire and forget”

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

Lab 7

Create a Delegated Form “Create AD User”

Lab 7

15 minutes

Create a Delegated Form “Create AD User”

Description

Create a new Delegated Form “Create AD User”

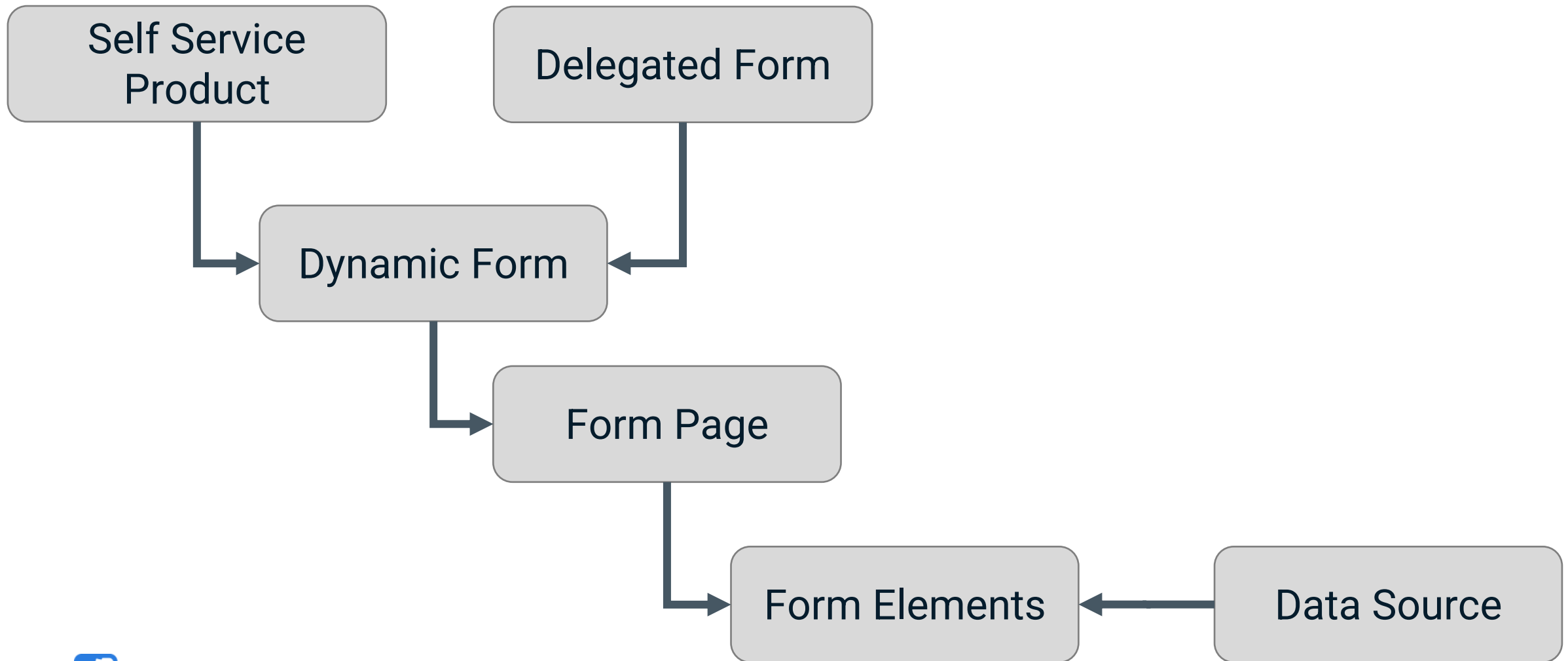
- Use existing Dynamic Form “Create AD User”
- Update the icon
- Configure Access Group

Testing

- Log in as Servicedesk employee, open the Delegated Form and test the Form UI

- No actions or tasks required

HelloID Service Automation Topology



HelloID Tasks / actions basics

HelloID Tasks / actions basics

- 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

Lab 8

Customer case

Lab 8

Customer case

10 minutes

Exam
topic

Customer:

Our servicedesk is getting a lot of authorization requests from employees and managers. Mainly for project folders access and password reset. Since we are using HelloID Service Automation, I would like to make use of this features.

Take some time to make a functional design

How to implement Service Automation

- Client needs?
 - Self Service Product
 - 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/en/requirements,-licensing,-policies/training/downloads.html>
 - HelloID training materials

Service Automation

Training day 2

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

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

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

PowerShell Variables

TYPE	NAME	VALUE	ACTIONS
st	firstname	{{form.givenname}}	 
st	lastname		
st	middlename		

PowerShell Variables

USE TEMPLATE

POWERSHELL SCRIPT

```
10
11
12 $ADUserParams = @{
13     Name           = $sAMAccountName
14     sAMAccountName = $sAMAccountName
15     AccountPassword = (ConvertTo-SecureString -AsPlainText
16     path           = $ou
17     Enabled        = $true
18     UserPrincipalName = $userPrincipalName
19     GivenName      = $firstname
20     Surname        = $lastname
21     DisplayName    = $displayName
22     Description    = "Created by HelloID Form"
23     Department     = $department
24     Title          = $title
25     AccountExpirationDate = $expDate
26 }
```

Products, scheduled tasks (and task data source)

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

```
1 Hid-Write-Status -Message "Result count: $resultCount" -Event Information
2 HID-Write-Summary -Message "Result count: $resultCount" -Event Information
3
4
5 Hid-Write-Status -Message "AD user [$sAMAccountName] created successfully" -Event Success
6 HID-Write-Summary -Message "AD user [$sAMAccountName] created successfully" -Event Success
7
8 HID-Write-Status -Message "Error searching AD users. Error: $($_.Exception.Message)" -Event Error
9 HID-Write-Summary -Message "Error searching AD users" -Event Failed
10
11
12
13 Hid-Add-TaskResult -ResultValue []
14 Hid-Add-TaskResult -ResultValue @{username = "demo01"; fullname = "Demo Account"; department = "Consultancy"}
```

Using variables – Delegated Forms and PowerShell data sources

No mapping needed

- Global variables `$ADUserUPNsuffix`
- Form variables `$form.lastname`
- Portal variables `$portalBaseUrl`
- Requester `$requester.userName`
- RequestedFor `$requestedFor.userName`

Delegated Forms and PowerShell data sources

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

```
1 Write-Information "Result count: $resultCount"
2 Write-Warning "No results found"
3 Write-Error "Error searching AD users. Error: $($_.Exception.Message)"
4
5 $Log = @{
6     Action = "CreateAccount" # optional. ENUM (undefined = default)
7     System = "ActiveDirectory" # optional (free format text)
8     Message = "Created account with username $userPrincipalName" # required (free format text)
9     IsError = $false # optional. Elastic reporting purposes only. (default = $false. $true = Executed action returned an error)
10    TargetDisplayName = $displayName # optional (free format text)
11    TargetIdentifier = $createdSID # optional (free format text)
12 }
13 Write-Information -Tags "Audit" -MessageData $Log
14
15 Write-Output @{username = "demo01"; fullname = "Demo account"; department = "Consultancy"}
```


Lab 9

Complete Delegated Form "Create AD User"

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

Lab 9

Example

```
1 # variables configured in form
2 $blnexpdate = $form.blnexpdate
3 $department = $form.department
4 $displayName = $form.naming.displayname
5 $expiredate = $form.expiredate
6 $firstname = $form.givename
7 $lastname = $form.lastname
8 $middlename = $form.middlename
9 $ou = $form.ou.Path
10 $password = $form.password
11 $sAMAccountName = $form.naming.samaccountname
12 $userPrincipalName = $form.naming.UserPrincipalName
13
14 try {
15     if($blnexpdate -ne 'true'){
16         $expDate = $null
17     } else {
18         $expDate = [datetime]$expiredate
19     }
20
21     Write-Information "Expiredate: $expDate"
22
23     $ADUserParams = @{
24         Name = $sAMAccountName
25         sAMAccountName = $sAMAccountName
26         AccountPassword = (ConvertTo-SecureString -AsPlainText $password -Force)
27         path = $ou
28         Enabled = $true
29         UserPrincipalName = $userPrincipalName
30         GivenName = $firstname
31         Surname = $lastname
32         DisplayName = $displayName
33         Description = "Created by HelloID Form"
34         Department = $department
35         Title = $title
36         AccountExpirationDate = $expDate
37     }
38
39     $tmp = New-ADUser @ADUserParams
40     Write-Information "AD user [$sAMAccountName] created successfully"
41 } catch {
42     Write-Error "Error creating AD user [$sAMAccountName]. Error: $($_.Exception.Message)"
43 }
```

Delegated Form

Import external template

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"

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

HelloID Data Sources

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 separate 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

1/2

- 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

2/2

Variables

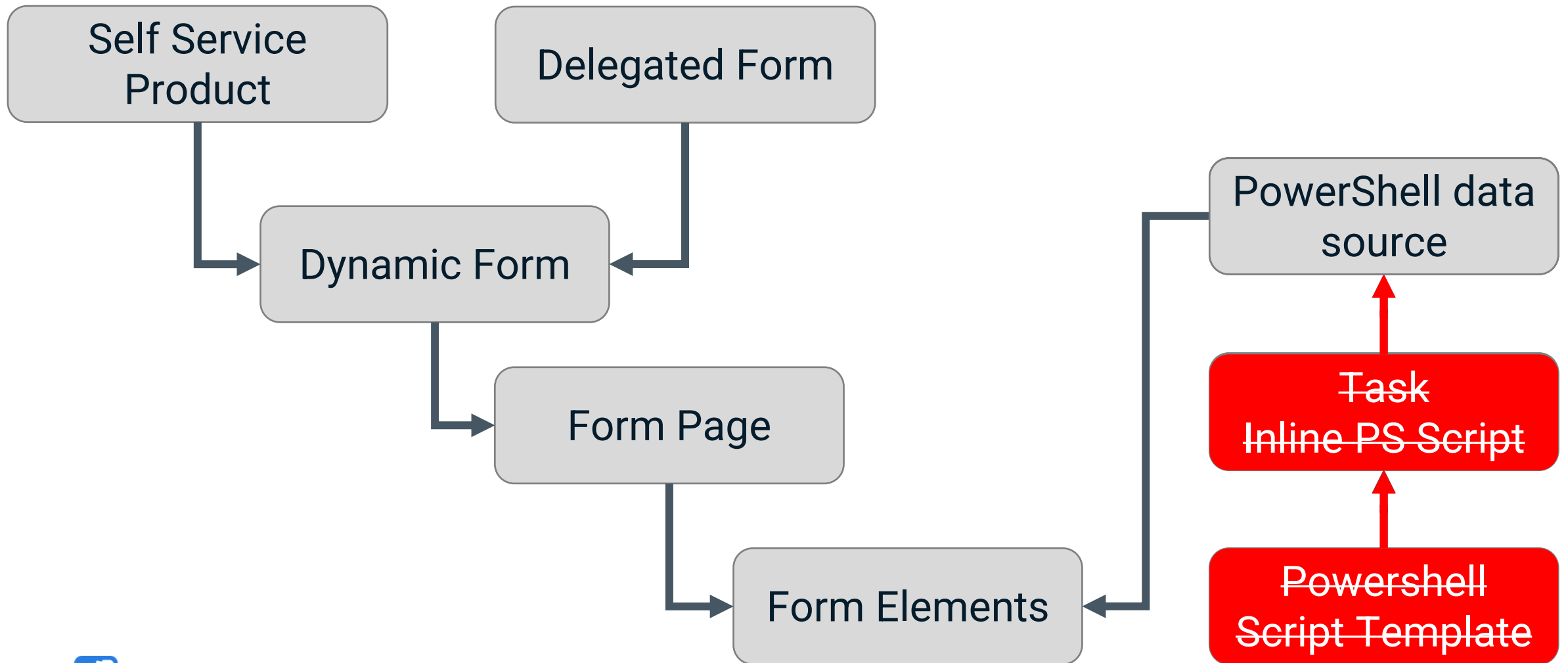
- Globale, user defined, variables → \$ADuserUPN
- Data source inputs → \$dataSource.
- Requester → \$requester.
- RequestedFor → \$requestedFor.

```
1 Write-Information $requester
2
3 try {
4     $searchValue = $dataSource.searchUser
5     $searchQuery = "$searchValue*"
6
7
8     $users = Get-ADUser -Filter {Name -like $searchQuery -or DisplayName -like $searchQuery} -properties displayName, UserPrincipalName, Department, Title
9     $users = $users | Sort-Object -Property DisplayName
10    $resultCount = @($users).Count
11    Write-Information "Result count: $resultCount"
12
13    if($resultCount -gt 0){
14        foreach($user in $users){
15            $returnObject = @{displayName=$user.displayName; UserPrincipalName=$user.UserPrincipalName; Department=$user.Department; Title=$user.Title;}
16            Write-Output $returnObject
17        }
18    }
19 } catch {
20     $msg = "Error searching AD user [$searchValue]. Error: $($_.Exception.Message)"
21     Write-Error $msg
22 }
```

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



Lab 11

Create a PowerShell data source for searching AD users

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

Lab 11

Example (very simple)

```
1 Write-Information $requester
2
3 try {
4     $searchValue = $dataSource.searchUser
5     $searchQuery = "$searchValue*"
6
7     $users = Get-ADUser -Filter {Name -like $searchQuery -or DisplayName -like $searchQuery} -properties displayName, UserPrincipalName, Department, Title
8     $users = $users | Sort-Object -Property DisplayName
9
10    if($resultCount -gt 0){
11        foreach($user in $users){
12            $returnObject = @{"displayName=$user.displayName; UserPrincipalName=$user.UserPrincipalName; Department=$user.Department; Title=$user.Title;"}
13            Write-Output $returnObject
14        }
15    }
16 } catch {
17     $msg = "Error searching AD user [$searchValue]. Error: $($_.Exception.Message)"
18     Write-Error $msg
19 }
```

Lab 12

Create a new delegated form for setting manager in AD

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 Form
- 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

Lab 13

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

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 Form
- Check your AD

Keep it simple

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

Lab 14

API time

Lab 14

?? minutes

API time

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>

How to implement Service Automation

- Client needs?
 - Self Service Product
 - 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/en/requirements,-licensing,-policies/training/downloads.html>
 - HelloID training materials