Fill out our survey and tell us how we did!
aka.ms/MIG/AdvancedQueriesWebinar

Microsoft Security

Building Advanced Queries for Users and Groups with Adaptive Policy Scopes

GEAR CAT TEAM

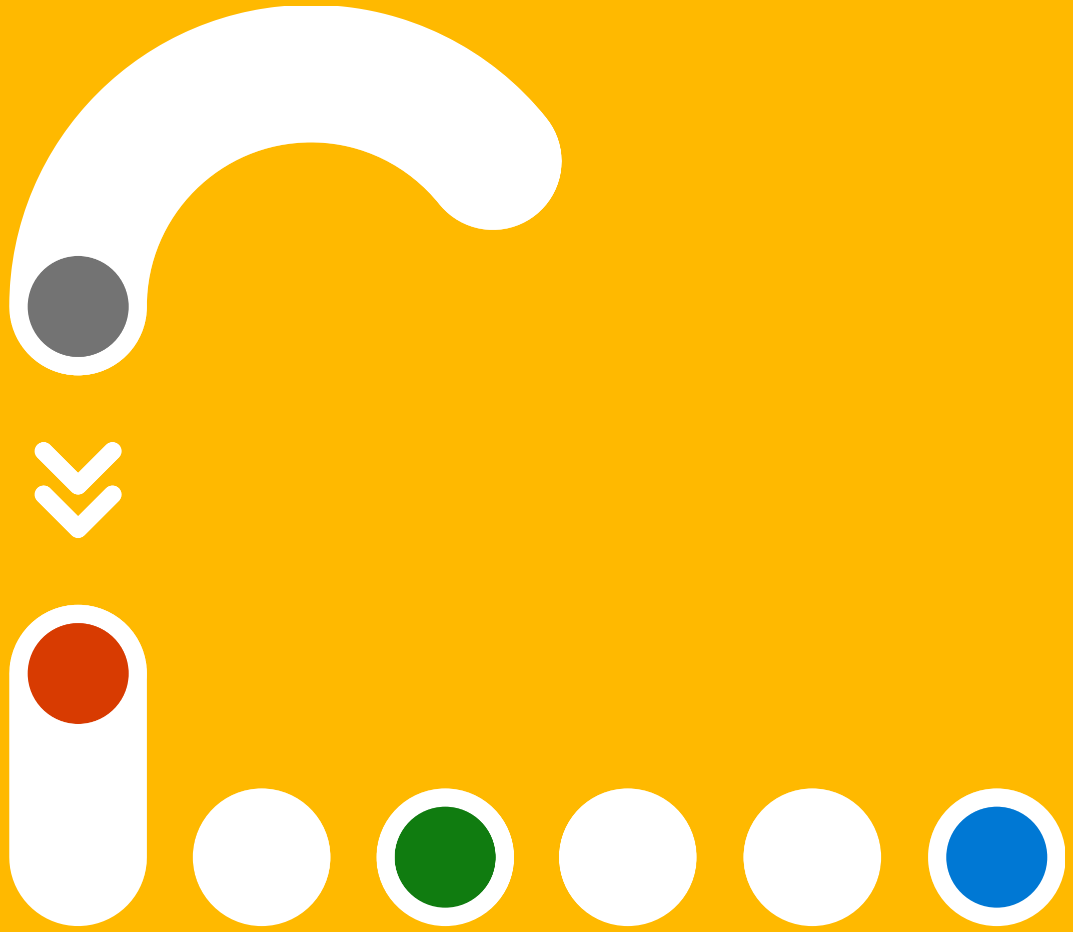
Brendon Lee
Sr Program Manager

Randall Galloway
Sr Program Manager

Agenda

- Adaptive policy scopes
- Simple query builder vs advanced query builder
- Best practices using OPATH to build advanced queries
- Validating advanced queries
- Demo – using advanced queries

Adaptive Policy Scopes



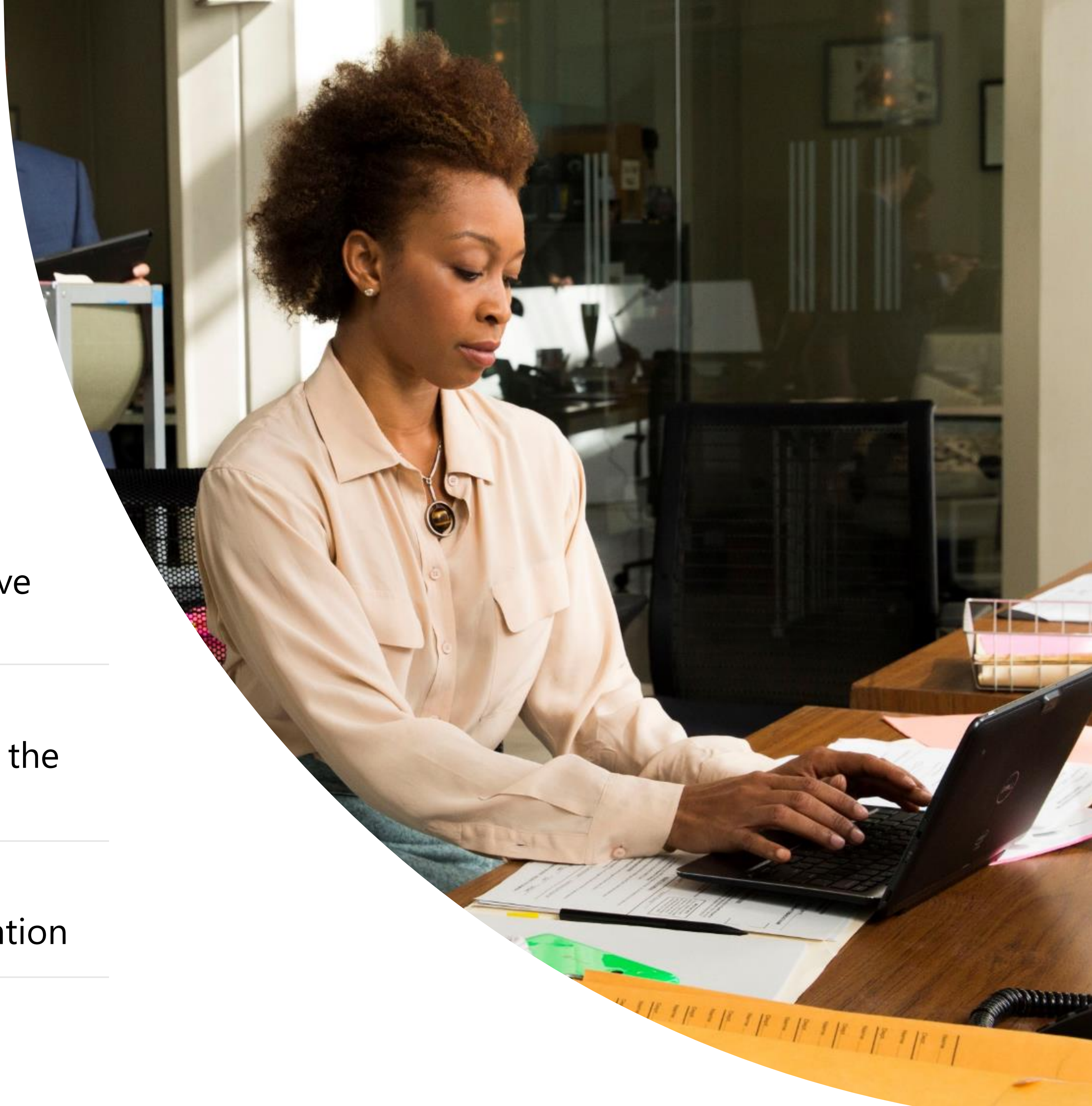
Adaptive policy scopes

Manage policy targeting with user, group, or site attributes

Automatic updates
Policies stay current as users join and leave roles

No more per-policy limits
Adaptive policy scopes are not subject to the previous include/exclude limits

New policy lookup
Understand which policy applies to a location



Adaptive Policy Scopes Introduction Webinar:

https://aka.ms/Adaptive-Webinar-EMEA-US

Simple query builder

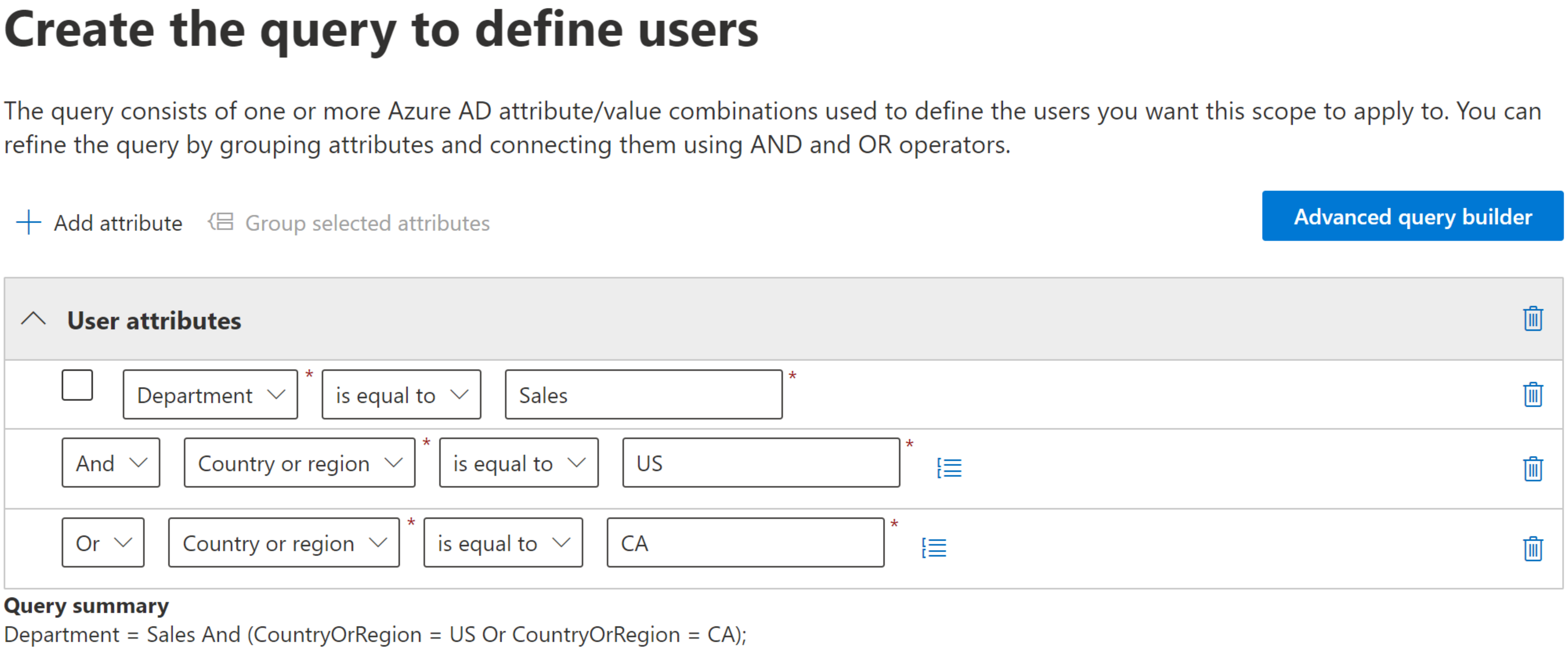
vs

Advanced query builder

with

User and M365 Group Scopes

Simple query builder



Create the query to define users

The query consists of one or more Azure AD attribute/value combinations used to define the users you want this scope to apply to. You can refine the query by grouping attributes and connecting them using AND and OR operators.

	Department *	is equal to	Sales *
And	Country or region *	is equal to	US *
Or	Country or region *	is equal to	CA *

Query summary
Department = Sales And (CountryOrRegion = US Or CountryOrRegion = CA);

Options with the simple query builder for users

Type	Locations supported	Available attributes
User	• Exchange email • OneDrive account • Teams chats • Teams private channel messages • Yammer user messages	• First Name • Last Name • Display Name • Job Title • Department • Office • Street Address • City • State or Province • Zip or Postal Code • Country or region • Email Address • Aliases • CustomAttribute1 - CustomAttribute15
Microsoft 365 Group	• Microsoft 365 Group (mailbox, site, or both) • Teams channel messages • Yammer Community messages	• Name • Display Name • Description • Email Addresses • Alias • CustomAttribute1 - CustomAttribute15

Operators

Logical Operators
- and
- or

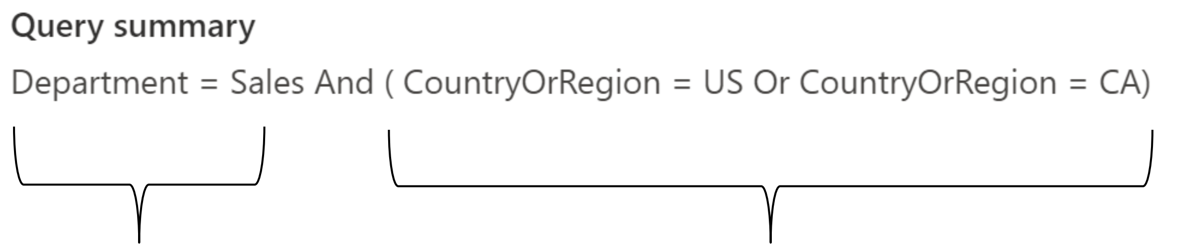
Comparison Operators
- Is equal to
- Is not equal to
- Starts with
- Not starts with

Grouping
- Limited grouping

Pros and cons: Simple query builder

Pros

- Very easy to quickly create a simple adaptive scope
- No knowledge of OPATH needed
- Easy to identify what the scope does in the GUI:



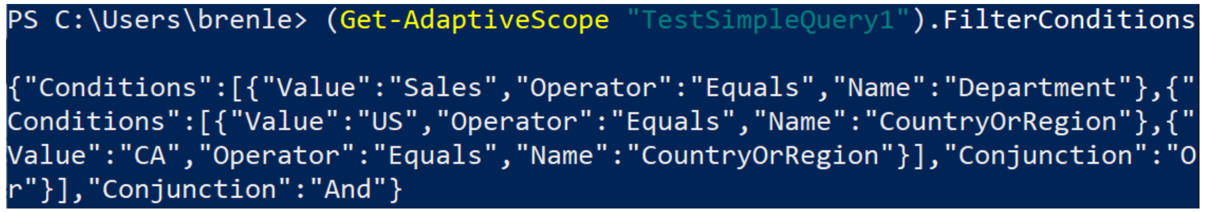
Query summary

Department = Sales And (CountryOrRegion = US Or CountryOrRegion = CA)

In Sales Dept **AND** in the United States **OR** Canada

Cons

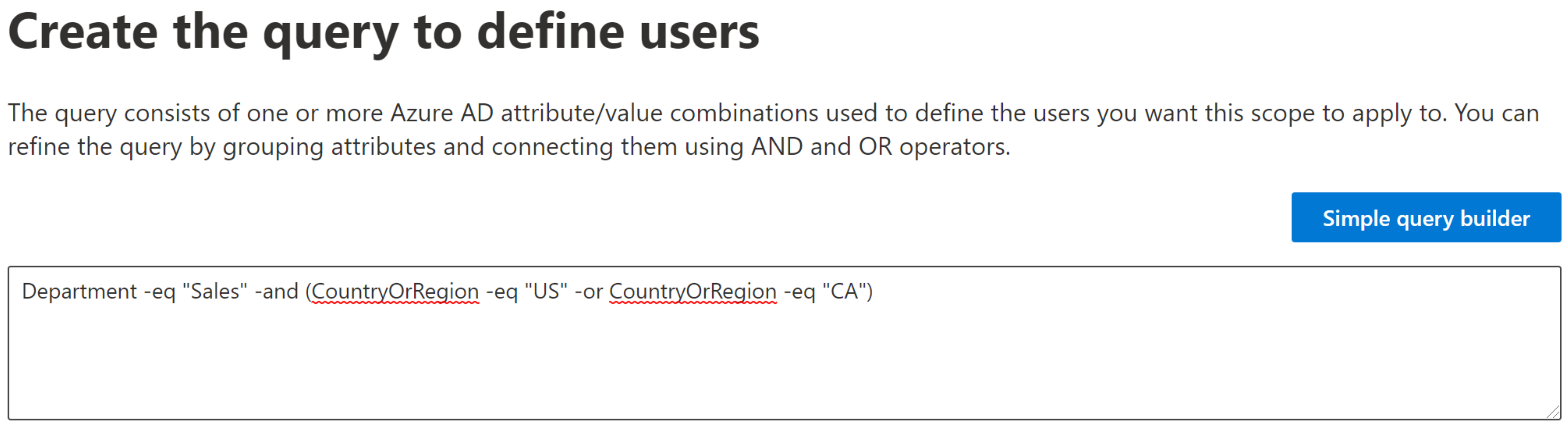
- Complicated queries can be difficult to configure
- No easy validation options
- Limited operators and grouping
- Not scalable
- Query stored in JSON, so can be difficult to quickly understand what a scope does in PowerShell:



```
PS C:\Users\brenle> (Get-AdaptiveScope "TestSimpleQuery1").FilterConditions

{"Conditions":[{"Value":"Sales","Operator":"Equals","Name":"Department"},{"
Conditions":[{"Value":"US","Operator":"Equals","Name":"CountryOrRegion"},{"
Value":"CA","Operator":"Equals","Name":"CountryOrRegion"}],"Conjunction":"O
r"}],"Conjunction":"And"}
```

Advanced query builder



Create the query to define users

The query consists of one or more Azure AD attribute/value combinations used to define the users you want this scope to apply to. You can refine the query by grouping attributes and connecting them using AND and OR operators.

```
Department -eq "Sales" -and (CountryOrRegion -eq "US" -or CountryOrRegion -eq "CA")
```

Pros and cons: Advanced query builder

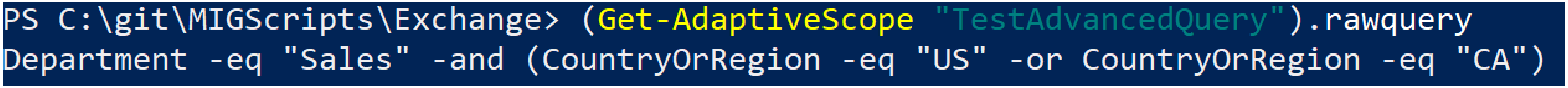
Pros

- Easy to pick up especially if you have previous experience with OPATH
- Much more flexibility in queries:
  - Support many properties that can be used with *Get-Mailbox* and *Get-Recipient* with –Filter parameter: aka.ms/opath-filter
  - Support all OPATH operators
- Easy to validate queries
- Easy to identify what the scope does both in the GUI and in PowerShell

Cons

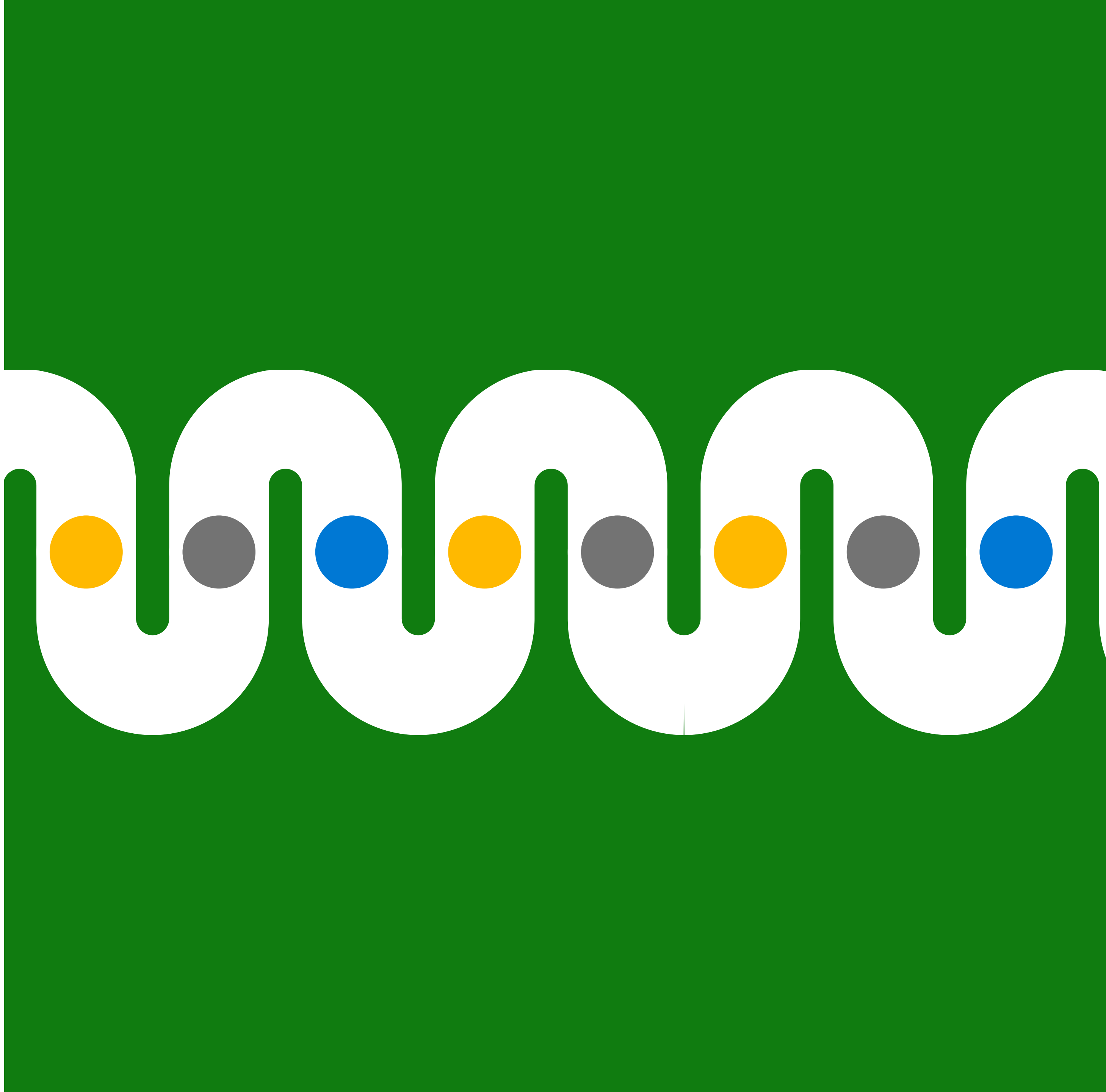
- Can be difficult to initially understand without previous experience
- Syntax rules can be tricky:
  - aka.ms/opath-syntax
  - This is what we are here to help with today!
- Currently no query validation built into the feature*

* We have a scripted solution though!

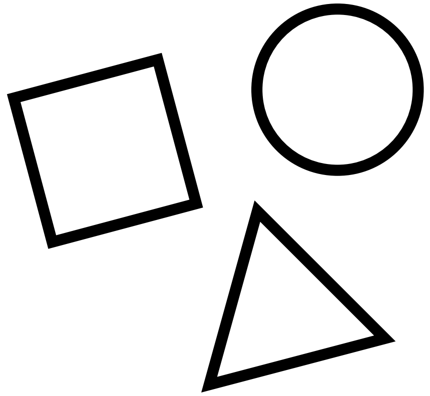


```
PS C:\git\MIGScripts\Exchange> (Get-AdaptiveScope "TestAdvancedQuery").rawquery
Department -eq "Sales" -and (CountryOrRegion -eq "US" -or CountryOrRegion -eq "CA")
```

Best practices using OPATH to build advanced queries



The basics



- You can **only** use OPATH properties that can *query mailboxes* using:
  - `Get-Mailbox -Filter`
  - `Get-Recipient -Filter`
- Some attributes are **not populated** for Groups because the AAD object **doesn't support it** – for example, *Department/Title/etc*
- Review properties at aka.ms/opath-filter
- Review OPATH syntax at aka.ms/opath-syntax

The logical operators

Logical Operators	Simple Query Builder	Description
`-and`	AND	Combine statements – all must be true
`-or`	OR	Separate statements – this OR that
`-not`	n/a	Negates following statement

- Carefully plan use of logical operators, especially with larger queries!

I want a policy that will…

- Apply to employees in **all countries** *except Russia, China, and Singapore*
- **Except** if they are *explicitly excluded* from retention

Will have **unintended results**:

```
    CountryOrRegion -ne “Russia” -or  CountryOrRegion -ne “China” -or CountryOrRegion
    -ne “Singapore” -or CustomAttribute1 -ne “ExemptedFromPolicy”
```

Correct:

```
    CountryOrRegion -ne “Russia” -and CountryOrRegion -ne “China” -and
    CountryOrRegion -ne “Singapore” -and CustomAttribute1 -ne “ExemptedFromPolicy”
```

The comparison operators

Comparison Operators	Simple Query Builder	Description
`-eq*`	Is equal to	Property value equals something
`-ne*`	Is not equal to	Property value does not equal something
`-like*`	Starts with*	Property value is like something (wildcard used)
`-notlike*`	Not starts with**	Property value is not like something (wildcard used)
`-lt`	n/a	Property value is less than something (number/date)
`-gt`	n/a	Property value is greater than something (number/date)

* Treated as 'contains' with multivalued properties

** OPATH Syntax rules restrict usage of wildcard

The gotchas

- Wildcard (*) is supported with `–like` / `-notlike`
  - Can **only** be a *suffix*
    - `Department –like “sales*”` ✓
    - `Department –like “*sales*”` ✗
- `$null` works, but Boolean **must** be represented as `True/False`
  - Title
  - LitigationHoldEnabled –eq ‘True’ ✓
  - LitigationHoldEnabled –eq $true ✗
- OPATH query must not be encased in quotes in Advanced Query Builder

The tips

- Use **parentheses** *only when needed* – e.g. to separate groups of statements with differing logical operators
  - `(Department -eq “Sales”) -and (CountryOrRegion -eq “US”)`
    - *This works – but is overly complex for no reason*
  - `(Department -eq “Sales” -and CountryOrRegion -eq “US”) -or Department -eq “Marketing”`
    - *This is a better use of parentheses*
- **VALIDATE**!

What can I do with OPATH that I can’t do with the simple query builder?

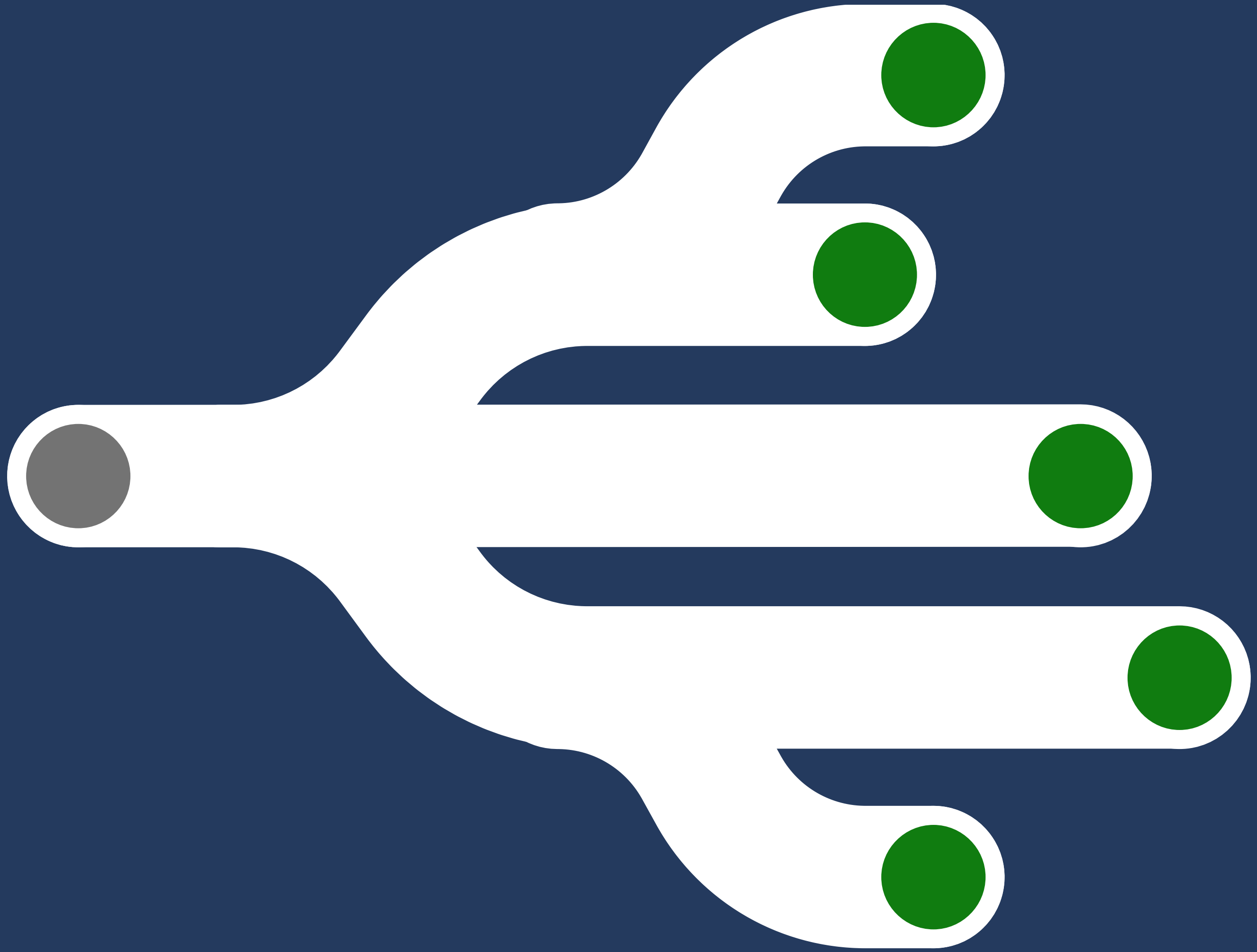
…A lot!

More supported OPATH properties at aka.ms/opath-filter…

Property Name & Description	In GUI?	Get-Mailbox	Get-Recipient	Value	-Filter Example
`Company` *Company Name (synced from AD/AAD)*	NO	✗	✓	• String *(wildcard accepted)* • $null	• `Company –eq 'Contoso'` • `Company –ne 'Contoso'` • `Company –like 'Contoso*'` • `Company –notlike 'Contoso*'`
`ExtensionCustomAttribute1-ExtensionCustomAttribute5` *Multi-valued custom properties (synced from AD/AAD)*	NO	✓	✓	• String *(wildcard accepted)* • $null	• `ExtensionCustomAttribute1 –eq 'ExcludeHolds'` • `ExtensionCustomAttribute2 –ne 'Intern'` • `ExtensionCustomAttribute4 –like 'Excluded*'` • `ExtensionCustomAttribute5 –notlike 'Mktg'`
`IsDirSynced` *Identifies synced objects (synced from AAD)*	NO	✓	✓	Boolean: • True • False	• `IsDirSynced –eq 'False'` • `IsDirSynced –ne 'False'`
`IsInactiveMailbox` *Identifies inactive mailboxes*	NO	✓	✗	Boolean: • True • False	• `IsInactiveMailbox –eq 'True'` • `IsInactiveMailbox –ne 'False'`
`LitigationHoldEnabled` *Identifies mailboxes on litigation hold*	NO	✓	✓	Boolean: • True • False	• `LitigationHoldEnabled –eq 'True'` • `LitigationHoldEnabled –ne 'True'`
`WhenCreated` *Date the user object was created (Synced from AD/AAD)*	NO	✓	✓	Date/Time (UTC)	• `WhenCreated -gt '10/22/2021 12:29:34 PM'` • `WhenCreated -lt '10/22/2021 12:29:34 PM'`

NOTE: As of now, `MemberOfGroup` and `Members` are not supported with Adaptive Policy Scopes

Validating advanced queries



Validation

- Currently, the advanced query builder **does not** have query validation *(Something we are working on)*
- You can **manually validate** using *Exchange Online PowerShell*
- The **cmdlet** you use to validate *depends on* the **properties** you want to use in the query
  - Use `–Filter` with whichever cmdlet you use
- Validation allows you to **review the expected results** of the query *immediately*

Validation basics

- **Remember** to use
  - `–IncludeInactiveMailbox` *with* `Get-Mailbox`
  - `-IncludeSoftDeletedRecipients` *with* `Get-Recipient`
- For **User scopes**, use
  - `-RecipientTypeDetails UserMailbox`
- For **Group scopes**, use
  - `-GroupMailbox` *with* `Get-Mailbox`
  - `-RecipientTypeDetails GroupMailbox` *with* `Get-Recipient`
- **Encase** your query in { } | **Watch** the quotes “””

Validation Example

Inactive mailboxes with LitigationHold Enabled

Property Name & Description	In GUI?	Get-Mailbox	Get-Recipient	Value	-Filter Example
`IsInactiveMailbox` *Identifies inactive mailboxes*	NO	✓	✗	Boolean: • True • False	• `IsInactiveMailbox -eq ‘True’` • `IsInactiveMailbox -ne ‘False’`
`LitigationHoldEnabled` *Identifies mailboxes on litigation hold*	NO	✓	✓	Boolean: • True • False	• `LitigationHoldEnabled -eq ‘True’` • `LitigationHoldEnabled -ne ‘True’`

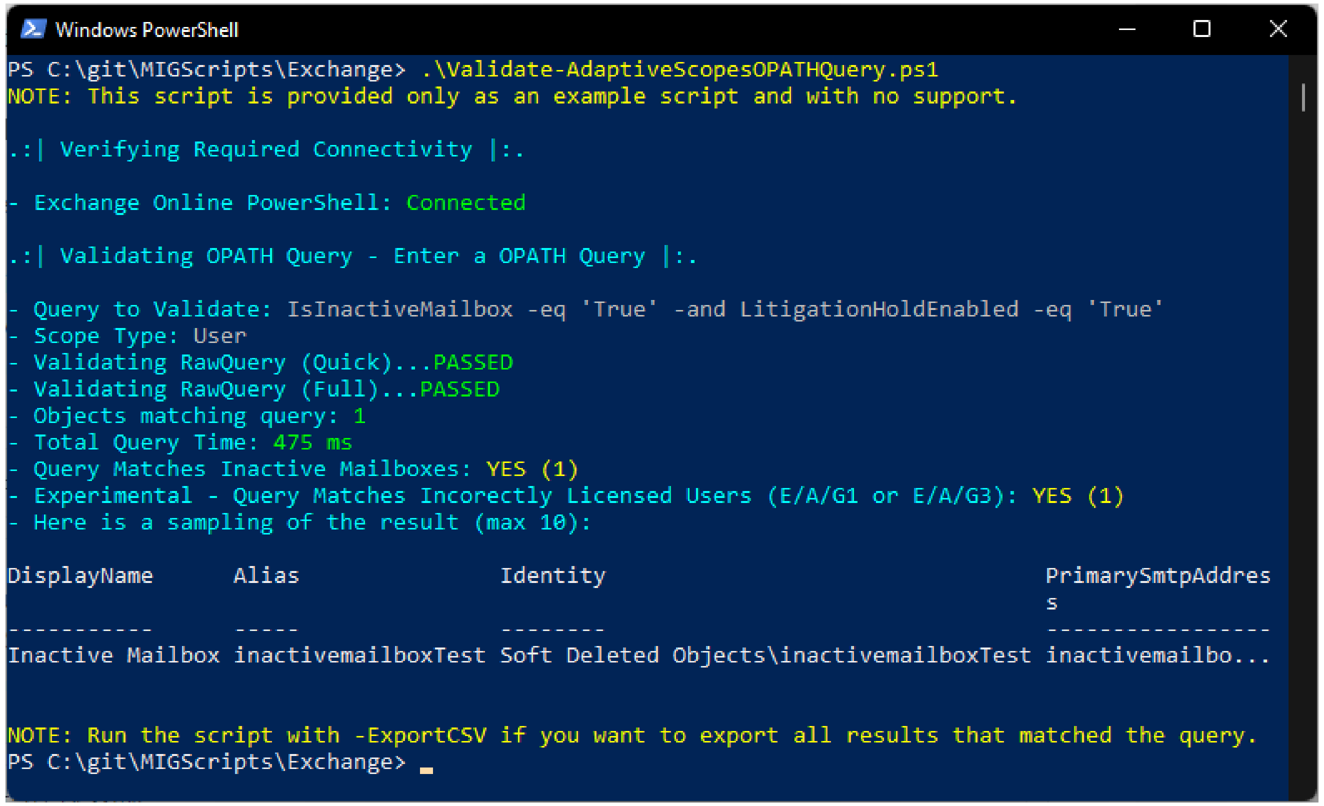
```
Get-Mailbox -RecipientTypeDetails UserMailbox -IncludeInactiveMailbox
-Filter {IsInactiveMailbox -eq “True” -and LitigationHoldEnabled -eq
“True”} -ResultSize Unlimited
```

TIP
If copying or pasting the query from somewhere else, you may want to replace the quotes “”/‘ ’

TIP
Use { } to surround the query to best replicate how it is input in the advanced query builder.

Validate-AdaptiveScopesOPATHQuery.ps1

- Alternatively, we’ve created a sample script that can be used to test OPATH queries for Adaptive Scopes
- A query can be tested from:
  - An easy to use GUI form
  - Via input parameter
  - From an existing Adaptive Scope
- Results can be output to CSV
- Download at aka.ms/ValidateAdaptiveScopeOPATH



```
PS C:\git\MIGScripts\Exchange> .\Validate-AdaptiveScopesOPATHQuery.ps1
NOTE: This script is provided only as an example script and with no support.

.:| Verifying Required Connectivity |:.

- Exchange Online PowerShell: Connected

.:| Validating OPATH Query - Enter a OPATH Query |:.

- Query to Validate: IsInactiveMailbox -eq 'True' -and LitigationHoldEnabled -eq 'True'
- Scope Type: User
- Validating RawQuery (Quick)...PASSED
- Validating RawQuery (Full)...PASSED
- Objects matching query: 1
- Total Query Time: 475 ms
- Query Matches Inactive Mailboxes: YES (1)
- Experimental - Query Matches Incorectly Licensed Users (E/A/G1 or E/A/G3): YES (1)
- Here is a sampling of the result (max 10):

DisplayName      Alias               Identity                                 PrimarySmtpAddres
                                                                              s
-----------      -----               --------                                 -----------------
Inactive Mailbox inactivemailboxTest Soft Deleted Objects\inactivemailboxTest inactivemailbo...


NOTE: Run the script with -ExportCSV if you want to export all results that matched the query.
PS C:\git\MIGScripts\Exchange> _
```

Where validation won't help

- Some OPATH Properties only work with either `Get-Mailbox` OR `Get-Recipient`

Property Name & Description	Get-Mailbox	Get-Recipient
`Company`	✗	✓
`IsInactiveMailbox`	✓	✗

- In these cases, PowerShell validation won't work
- But, CAN be combined in Advanced query builder

Demo

The Scenario

- Must replace existing org-wide policy
- Need to determine **policy inclusion/exclusion** based on **business unit**, **cost center**, **license** and **termination status**

Business unit	Cost Center
CORP	**110**233, **110**234, **119**229
EXEC	**110**230
~~**RETAIL**~~	~~111330~~

- CORP policy must **only** apply to **CORP business unit** *(also consider if new cost centers are added within range)*
- Need **CapStone** approach for **Executives**
- Need **General 2 year retain & delete** for **departed employees**

  - Will use **ExtensionCustomAttribute** because it is a multi-value property
  - Will use **PersistedCapabilities** for license
  - Will use **IsInactiveMailbox** for term status

The Plan

- Must replace existing org-wide policy
- Need to determine **policy inclusion/exclusion** based on **business unit**, **cost center**, **license** and **termination status**

Business unit	Cost Center
CORP	**110**233, **110**234, **119**229
EXEC	**110**230
~~RETAIL~~	~~111330~~

Create 3 scopes:

- *Corp Scope*:
  - CORP business unit
  - Cost centers 110*/119*
  - Active users only
  - E5 Only
- *Exec Scope (CapStone approach)*:
  - EXEC business unit
  - Inactive/Active
  - E5 Only
- *Inactive Scope*:
  - Any inactive/departed user

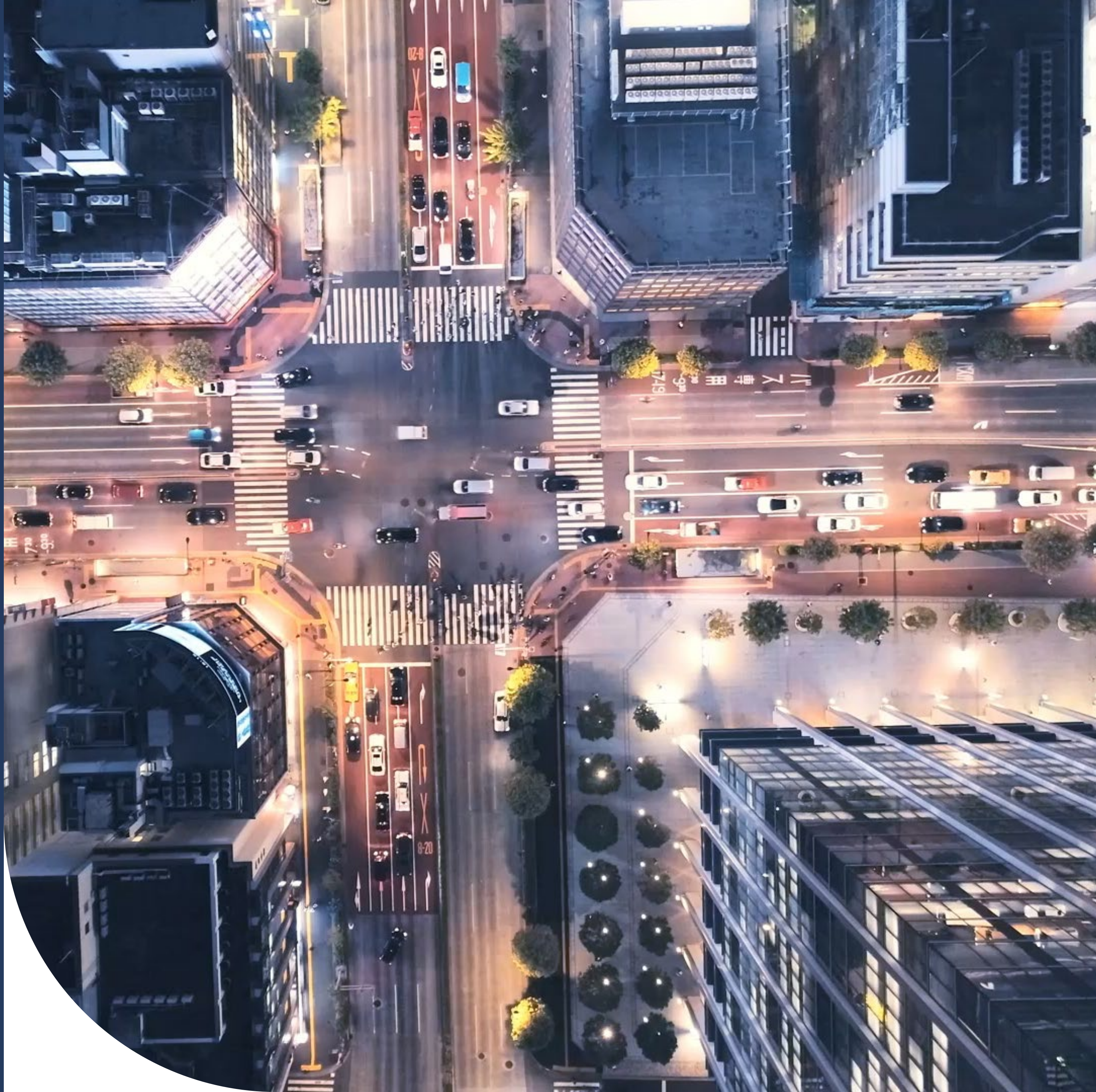
Microsoft Security

Save the Date

Building Advanced Queries for SharePoint Sites with Adaptive Policy Scopes

December 9th, 2021
16:00 GMT / 8:00AM PST

aka.ms/AdaptiveScopes-AdvancedQueries



Resources

Deployment Guide
https://aka.ms/MIG/Deployment

Webinars
https://aka.ms/MIG/Webinars

Blog
https://aka.ms/MIG/Blog

Documentation
https://aka.ms/MIG/Documentation

Videos
https://aka.ms/MIG/Videos

Adaptive Scopes Public Preview Release Webinar
https://aka.ms/Adaptive-Webinar-EMEA-US

Survey (for this webinar)
https://aka.ms/MIG/AdvancedQueriesWebinar

Thank you

Fill out our survey and tell us how we did!
aka.ms/MIG/AdvancedQueriesWebinar

Microsoft Security