BC/NAV TECH DAYS 2022

mibuso.com

10 YEAR ANNIVERSARY

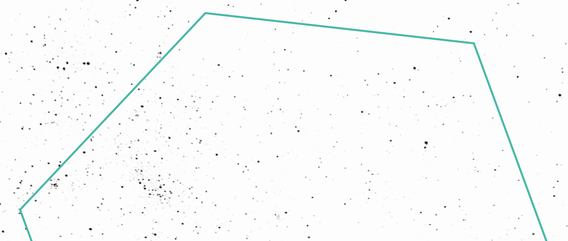
www.bctechdays.com

Permissions revisited

Darrick Joo, Jens Møller-Pedersen
MICROSOFT

Agenda

- Introduction
- Composing permission sets
- Exclusion
  - How it works
  - Examples
- New UI for creating permission sets
- InherentPermissions / InherentEntitlements
- What's Next

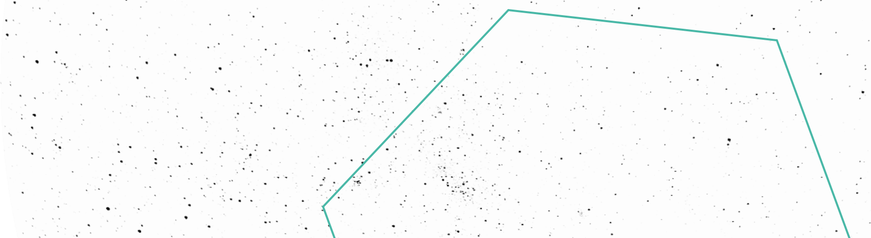


Introduction

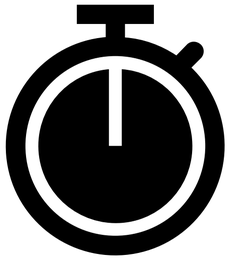
Business Central Permission System

Testing	• Test Permissions
Administrator	• Permissions
Constraints	• Read-only access
IP Owner	• Entitlements • Licenses (on-premise)

User needs access in each of these layers



What are we looking to achieve?



Spend less time setting up permissions

Understand what you have set up

Reduce maintenance burden

Well defined, understandable building blocks

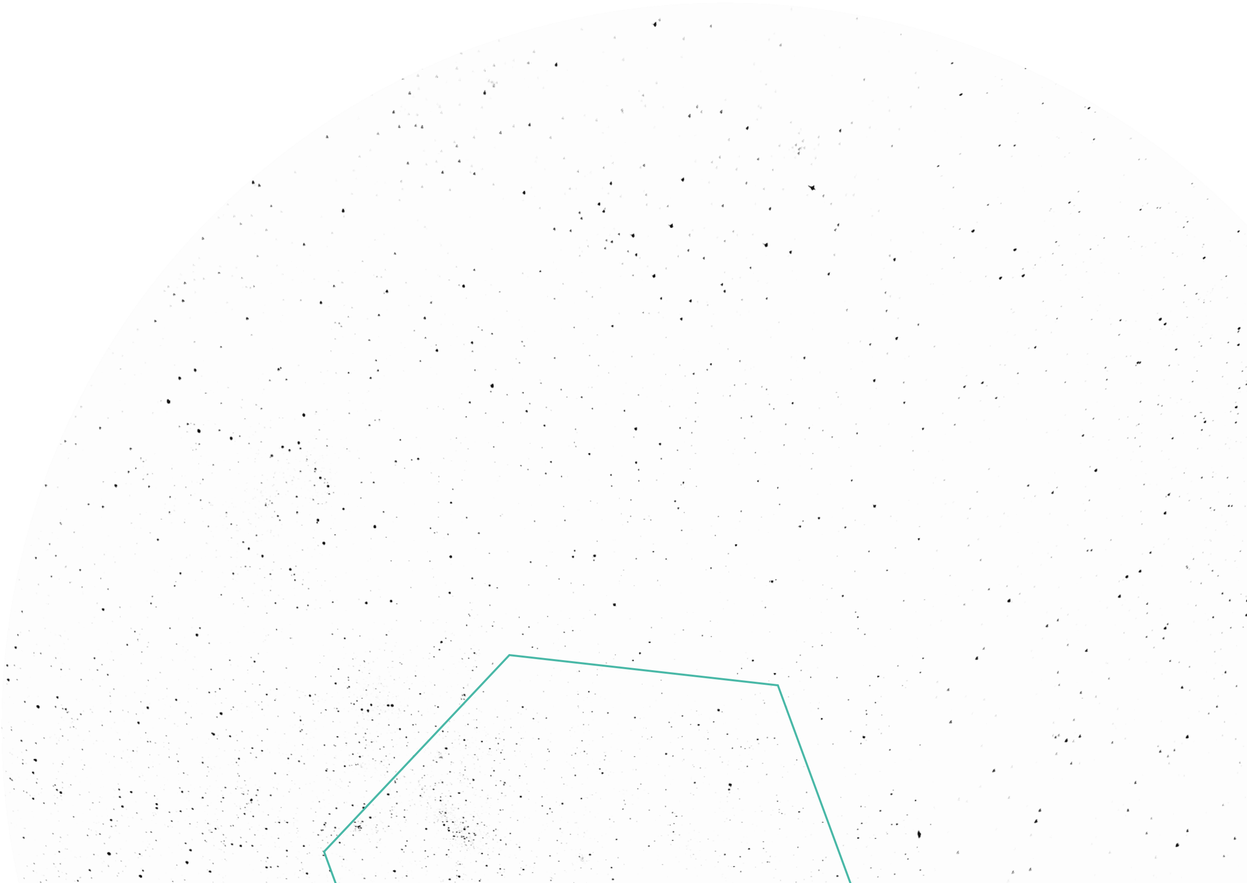
End users work with permission sets – not permissions

Permission set explains what it does

- It contains permissions needed to execute a task or perform a duty

The administrator grants access to

- The duties that users in a role perform
- **Not** to the program elements that users must use



Reduce maintenance burden

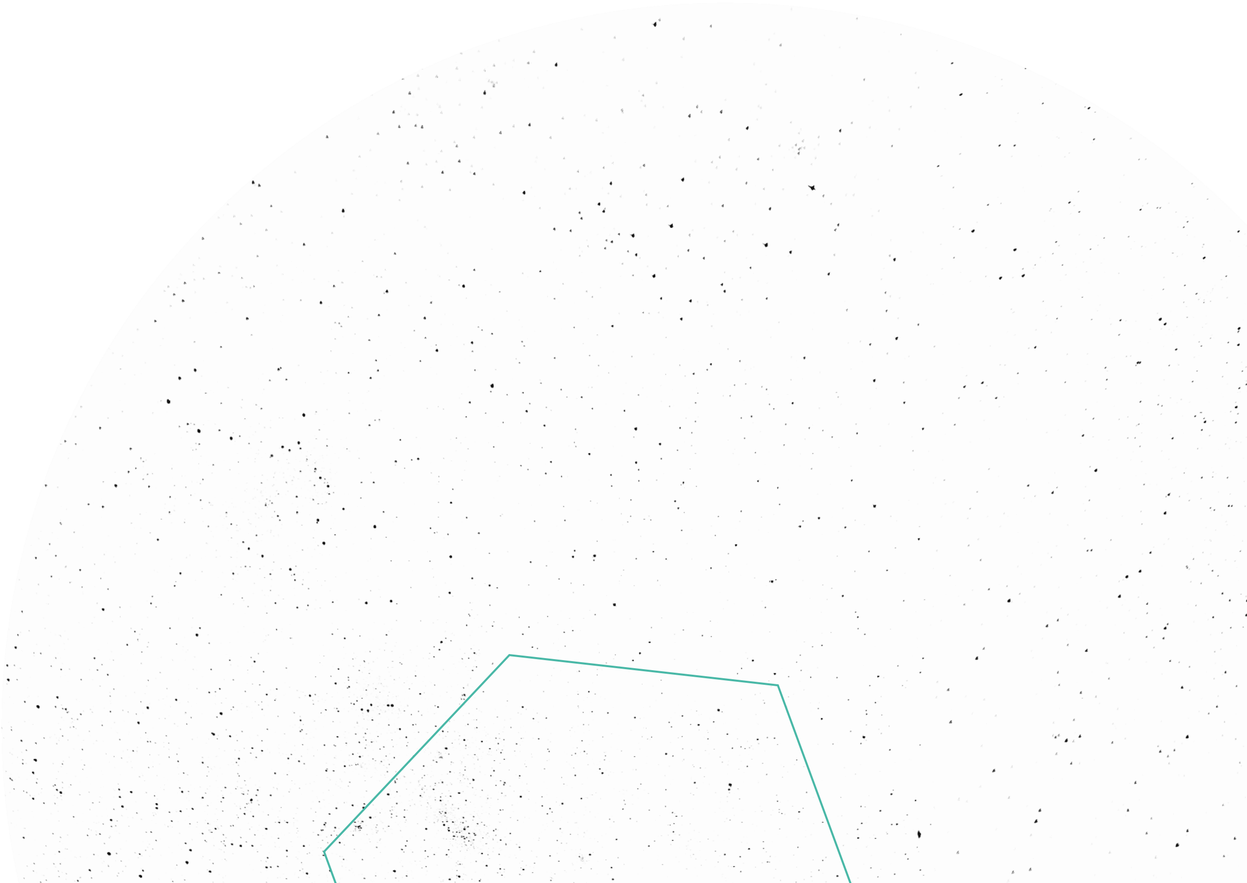
Reduce friction when upgrading to new versions

- No need to maintain your own copies

Distribute permissions with your extensions

Automatically assign roles to new users

Limit impact of missing permissions

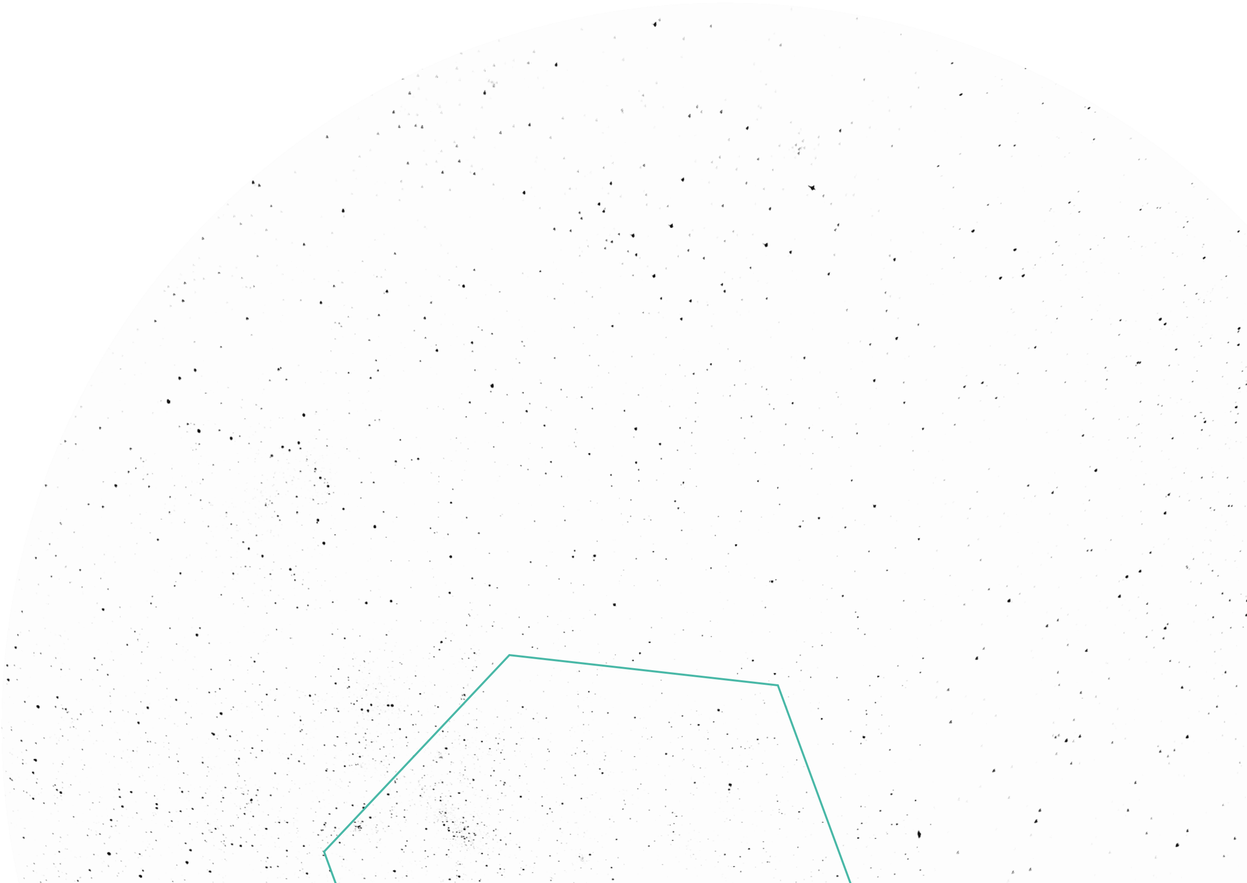


Simplify permission sets

Reduce size of permission sets

Remove ‘odd’ permissions for

- Initialization
- Helper methods

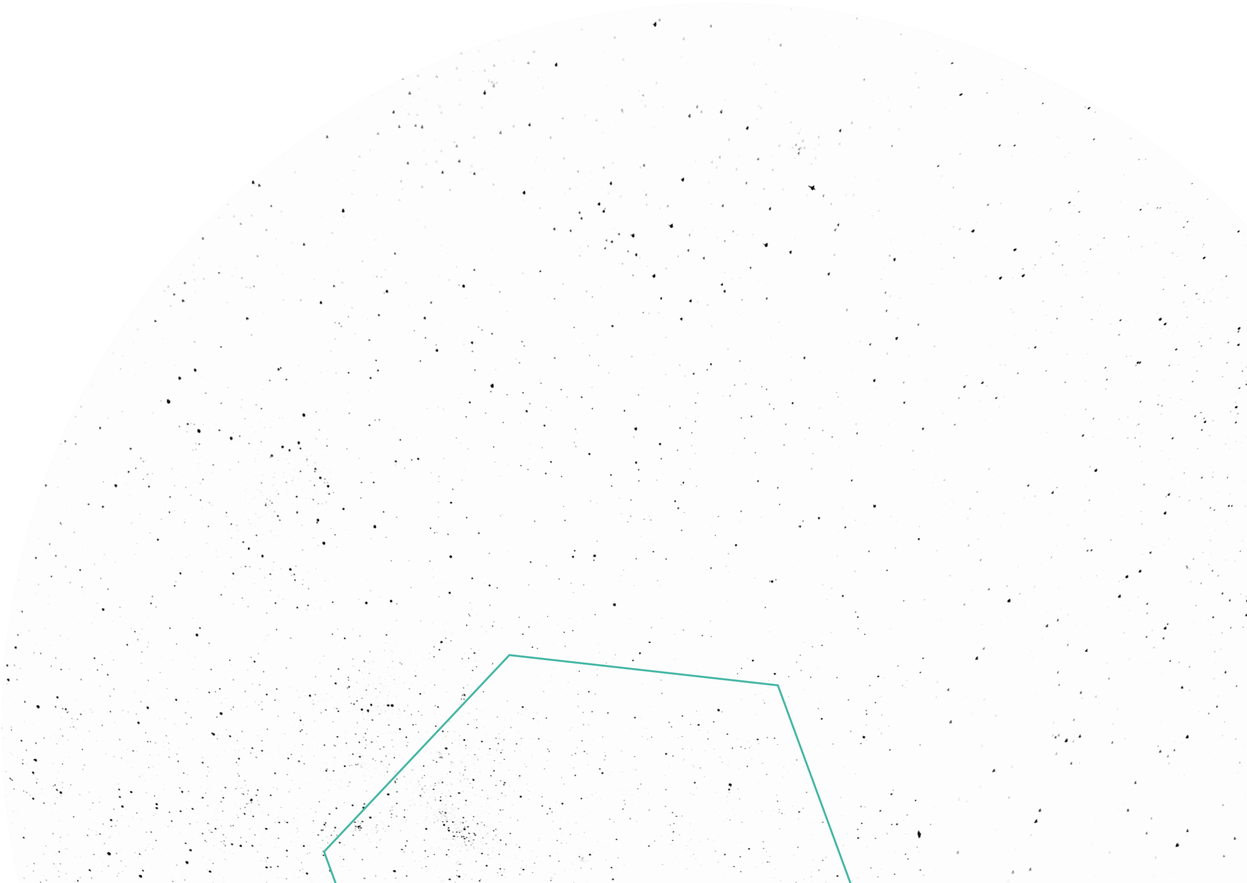


Greater flexibility

Create the granularity you need

Explicit adjustment of default roles and permission sets

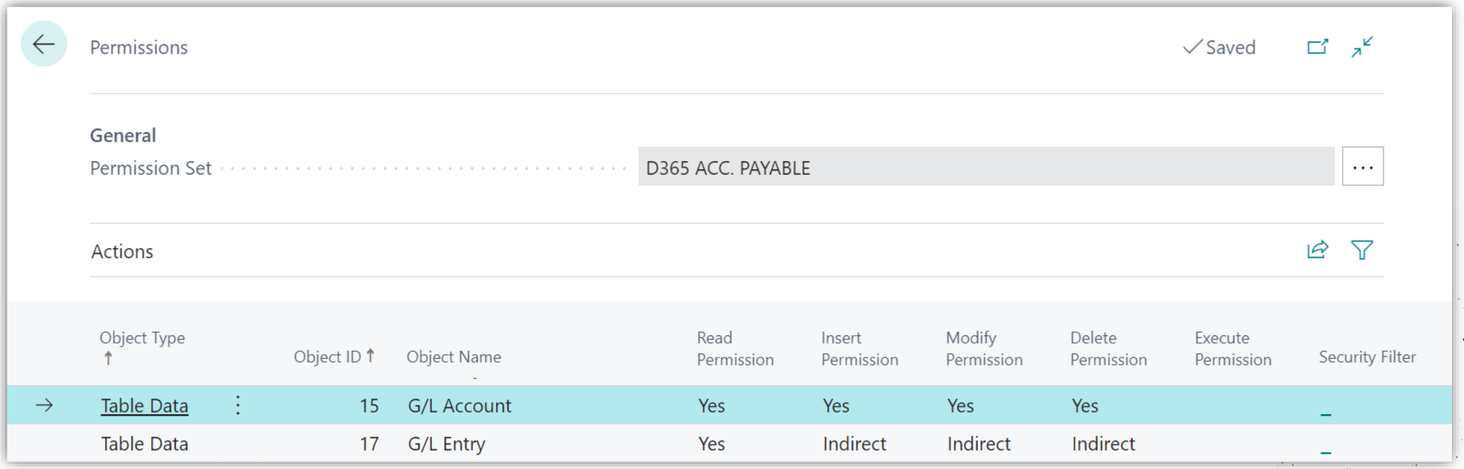
Reuse across tenants using Apps



Composing permission sets

Permission set

- Contains object permissions

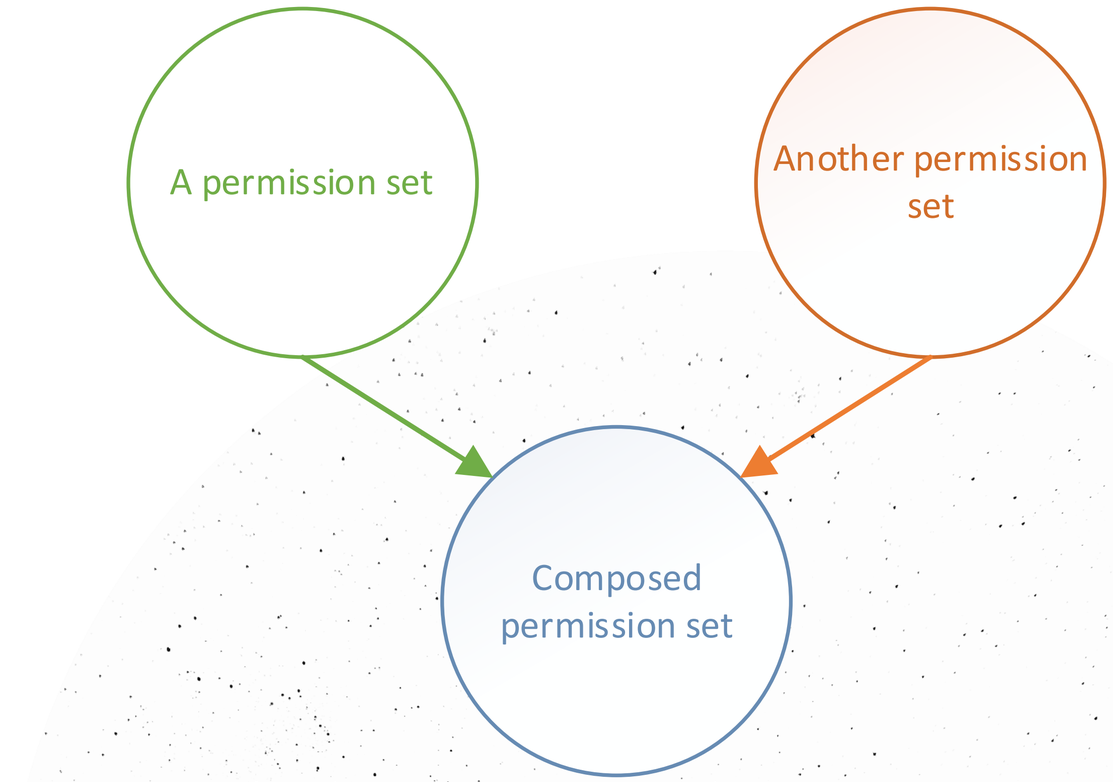



```
permissionset 681 "D365 JOURNALS, POST"
{
    Assignable = true;

    Permissions = tabledata "G/L Entry" = Rimd,
                  tabledata "Item Register" = Rimd,
```

Composed permission set

- Includes other permission sets permissions

"Composed permission set" permissions =
"A permission set" permissions +
"Another permission set" permissions +
Permissions



Composed permission set (example)

- Includes other permission sets permissions

“D365 BUS FULL ACCESS” permissions =
“D365 BASIC” permissions +
“D365 HR, EDIT” permissions +
Permissions

“D365 BUS PREMIUM” permissions =
“D365 BUS FULL ACCESS” permissions +
“D365 MFG, EDIT” permissions +
Permissions

How to define a composed permission set?

BC/NAV TECH DAYS 2022
mibuso.com

In an extension

Added in 2021 Wave 1

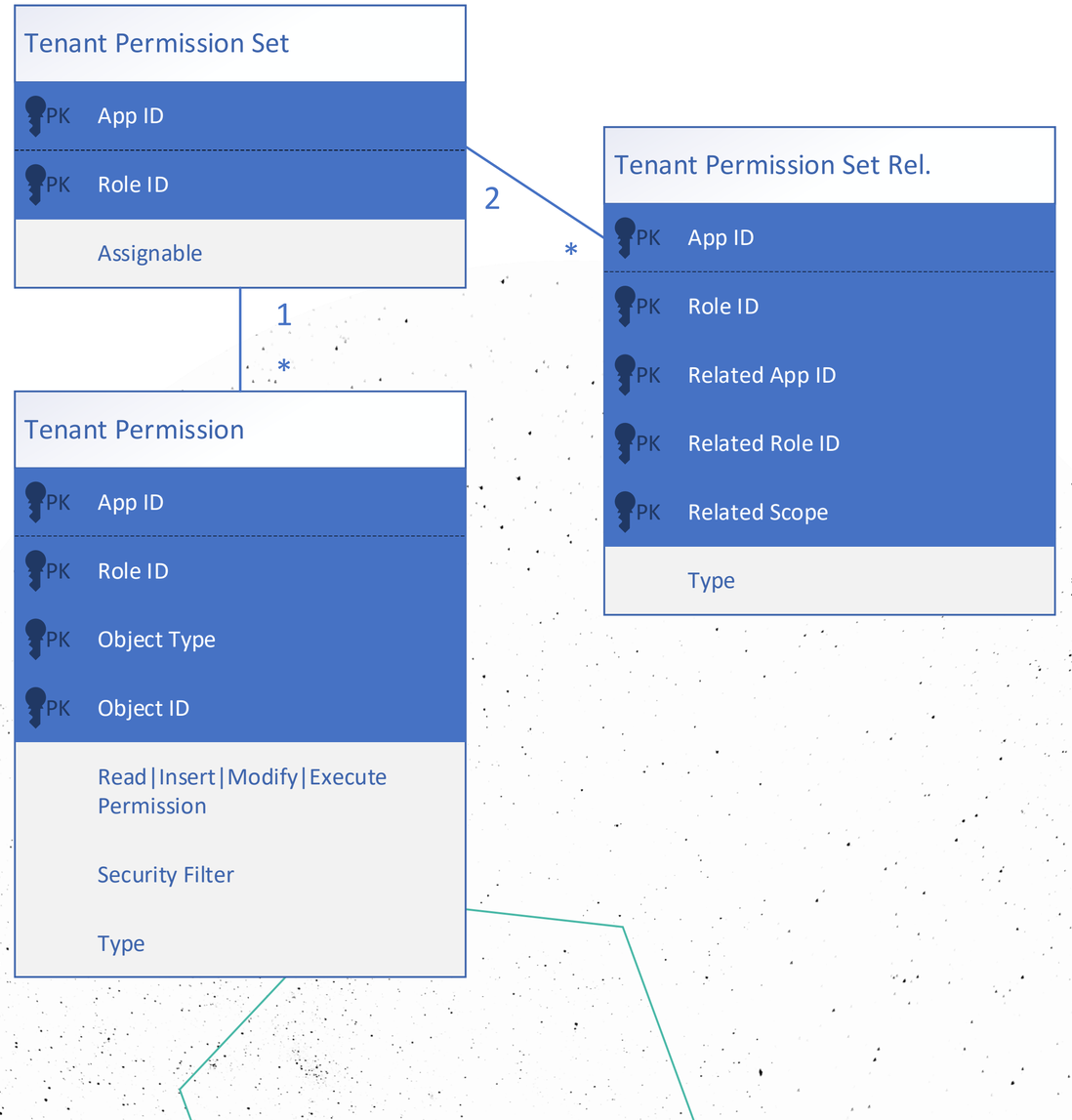
```
permissionset 959 "D365 BUS FULL ACCESS"
{
    Assignable = true;

    IncludedPermissionSets = "Company - Edit",
                             "D365 BASIC",
                             "D365 ADCS, EDIT";

    Permissions = tabledata "Item Availability by Date" = RIM,
                  tabledata "Item Availability Line" = RIM,
                  tabledata "Item Budget Entry" = RIMD,
                  tabledata "Item Budget Name" = RIMD,
                  tabledata "Item Category" = RIMD,
```

Define a composed permission set

New in 2022 Wave 1

- New system table:
  "Tenant Permission Set Rel."
  - Defines relations between permission sets

- Hierarchy is retained at runtime

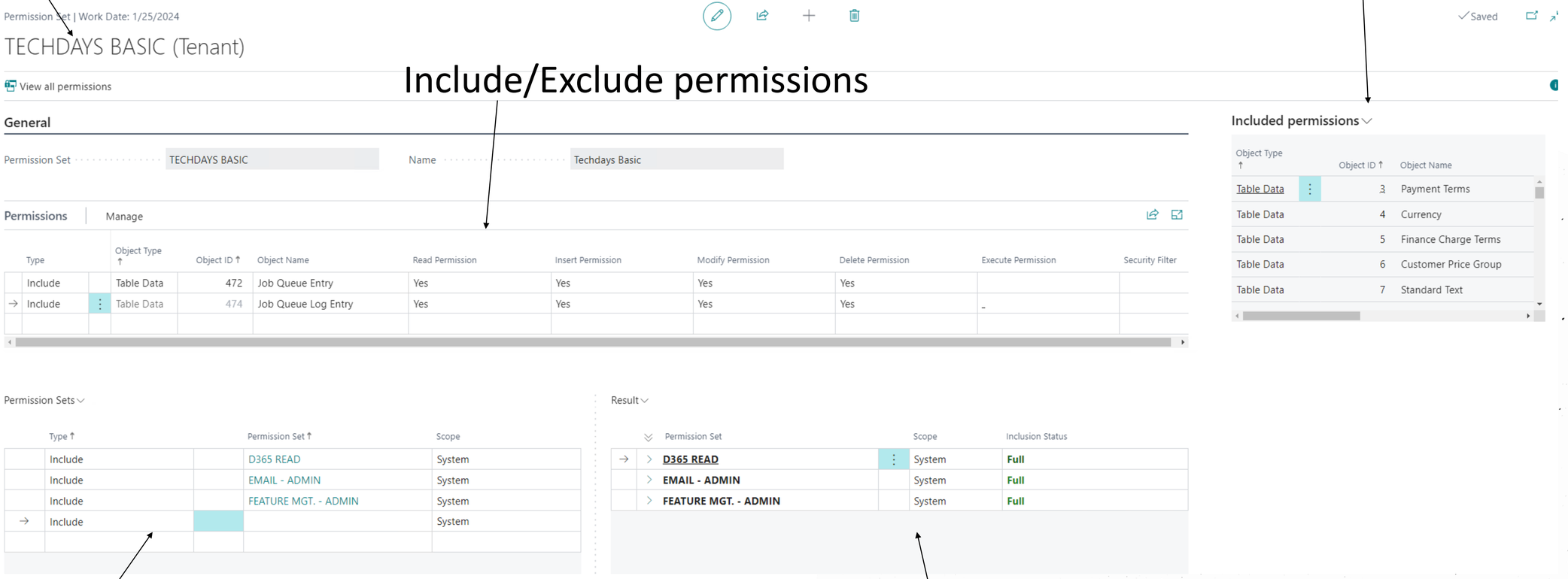


Define a composed permission set in the UI

PS Name (Type)

Included permissions for the currently selected PS

Include/Exclude permissions

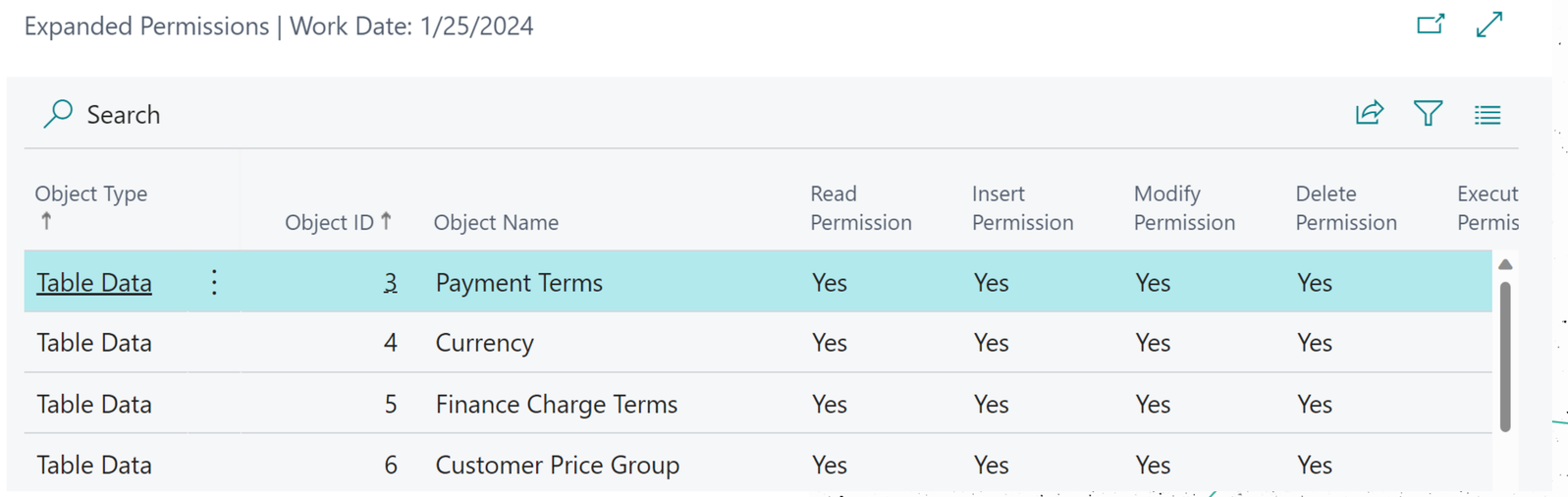


Include permission sets

Result of included permission sets and hierarchy

How to see the result of the composition?

- New system table “Expanded Permissions”
  - Compute on the fly the flattened list of permissions in a permission set
  - Used to compute user permissions and for “Effective permissions” page



Benefits

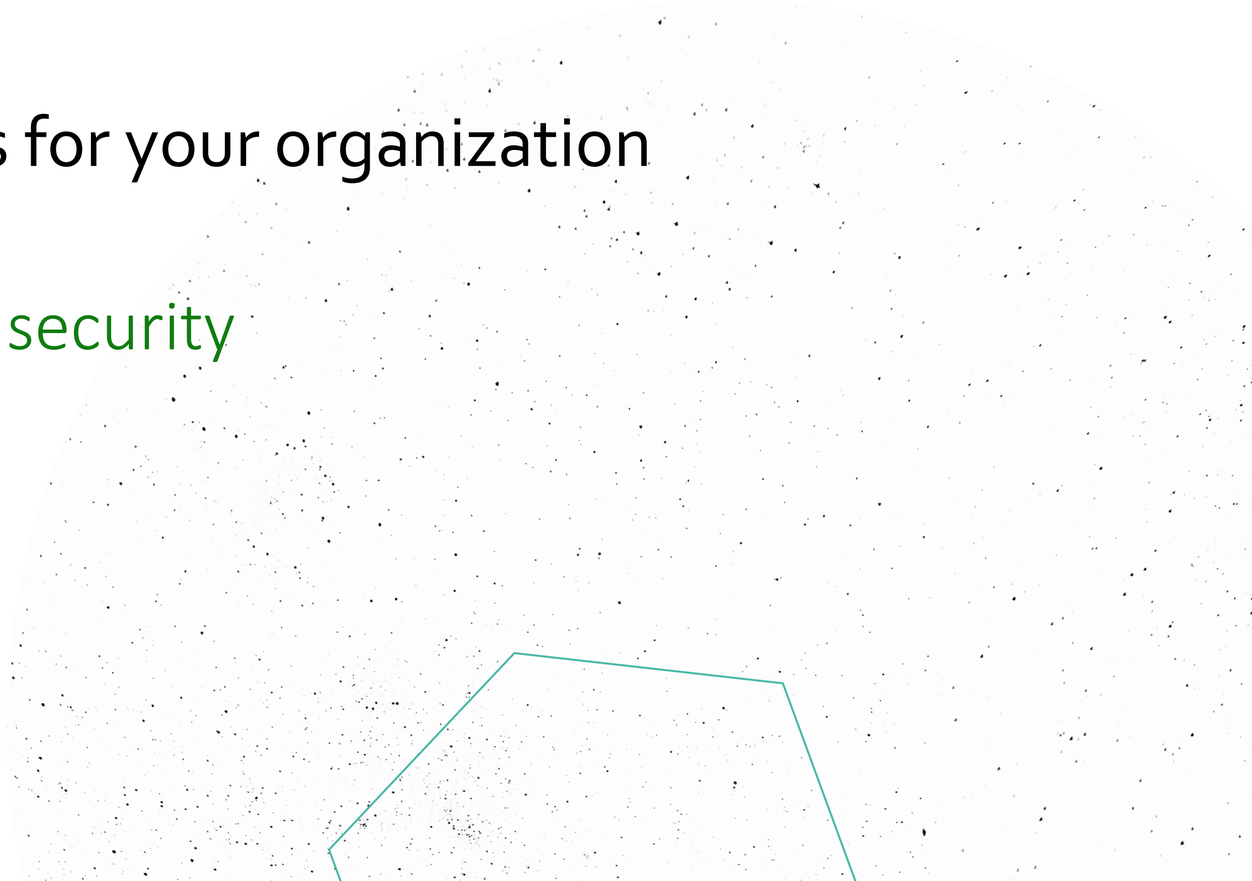
Partners

- Create fine grained hierarchies that makes sense for your line of business

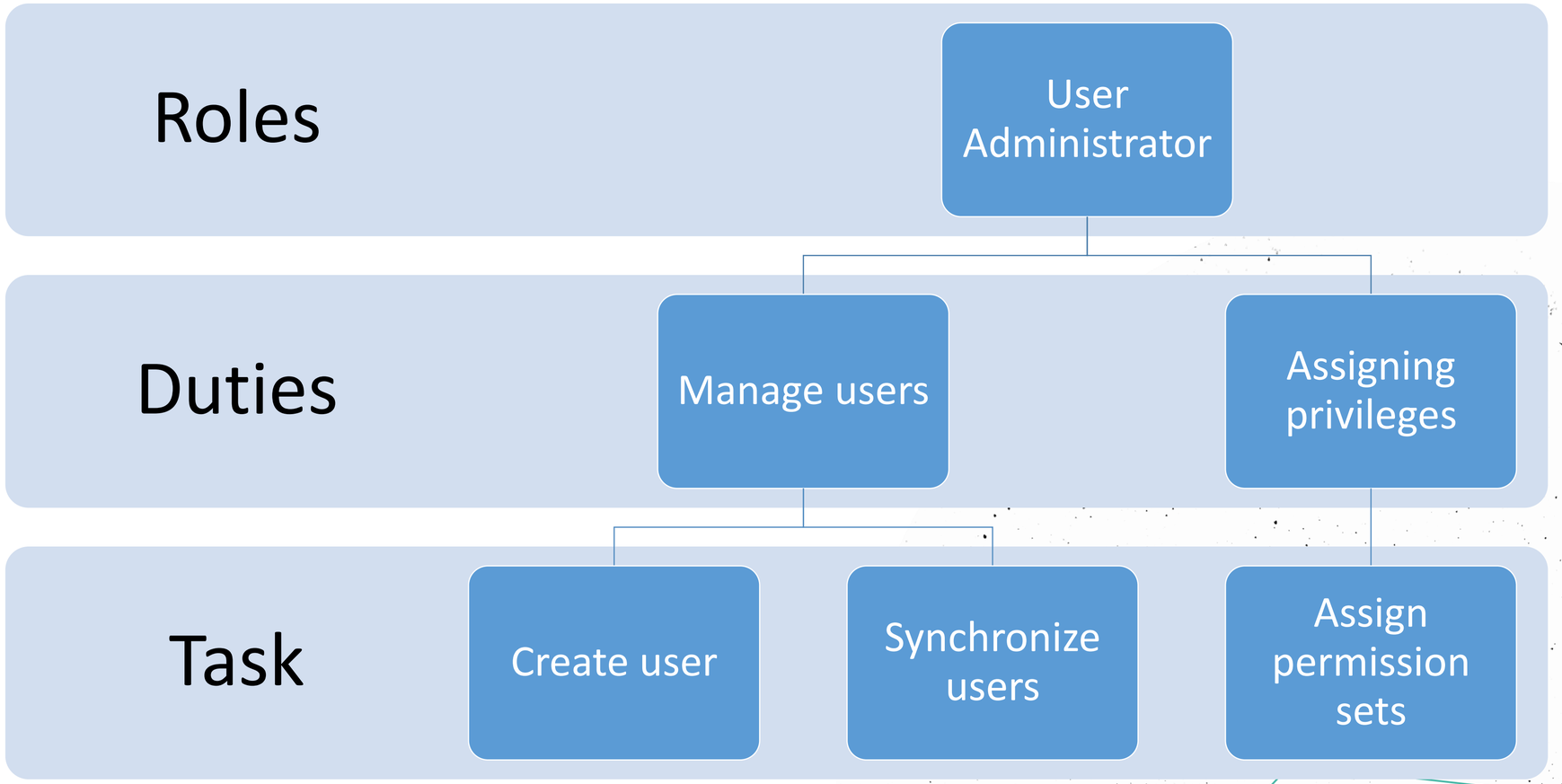
Administrators

- Create hierarchies that makes senses for your organization

You decide how to structure your system security



Sample hierarchy



Exclusion

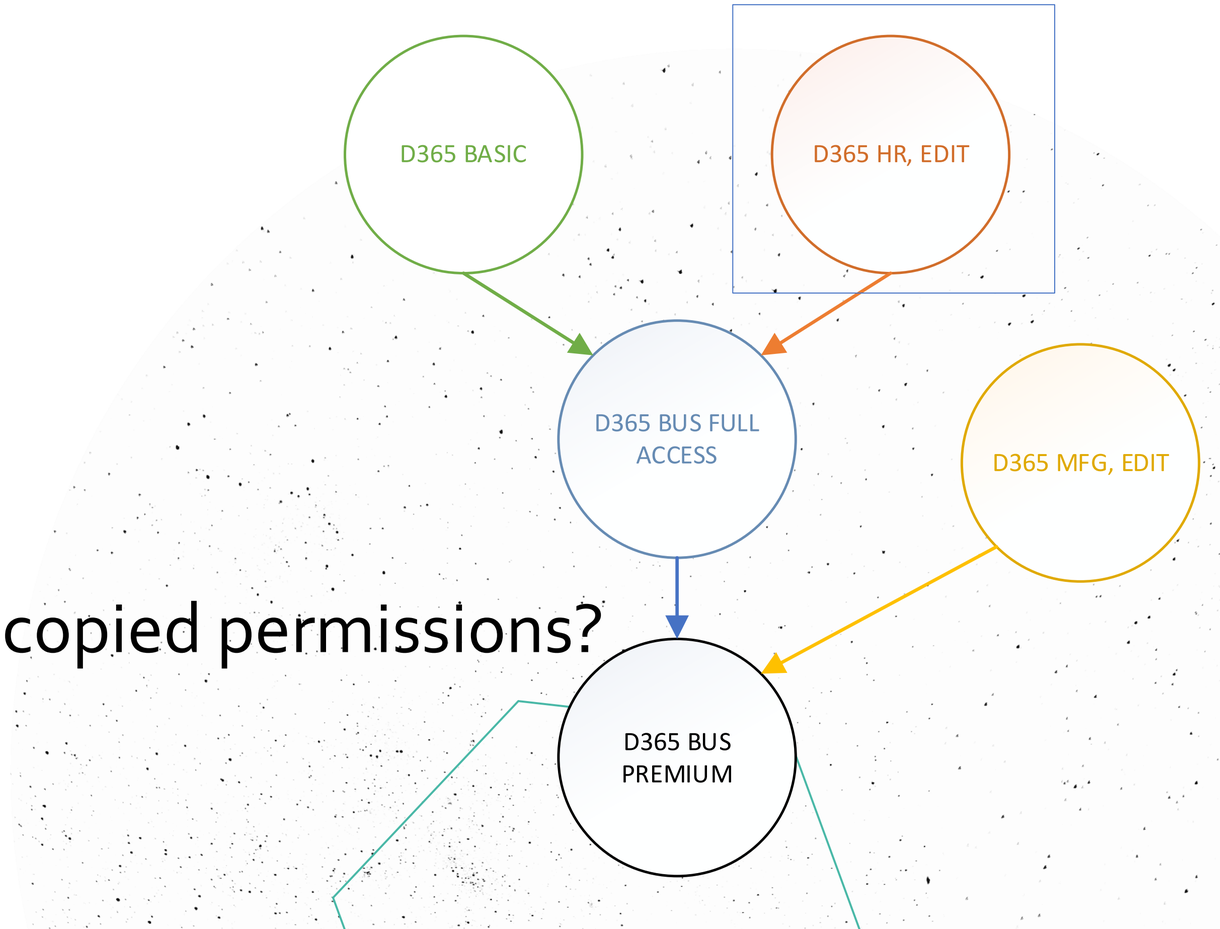
But what if it’s not right for you?

What if the hierarchies defined by Microsoft or ISVs are not fine grained enough?

And include too many permissions?

“Copy permission set”?

And end-up synchronizing forever the copied permissions?

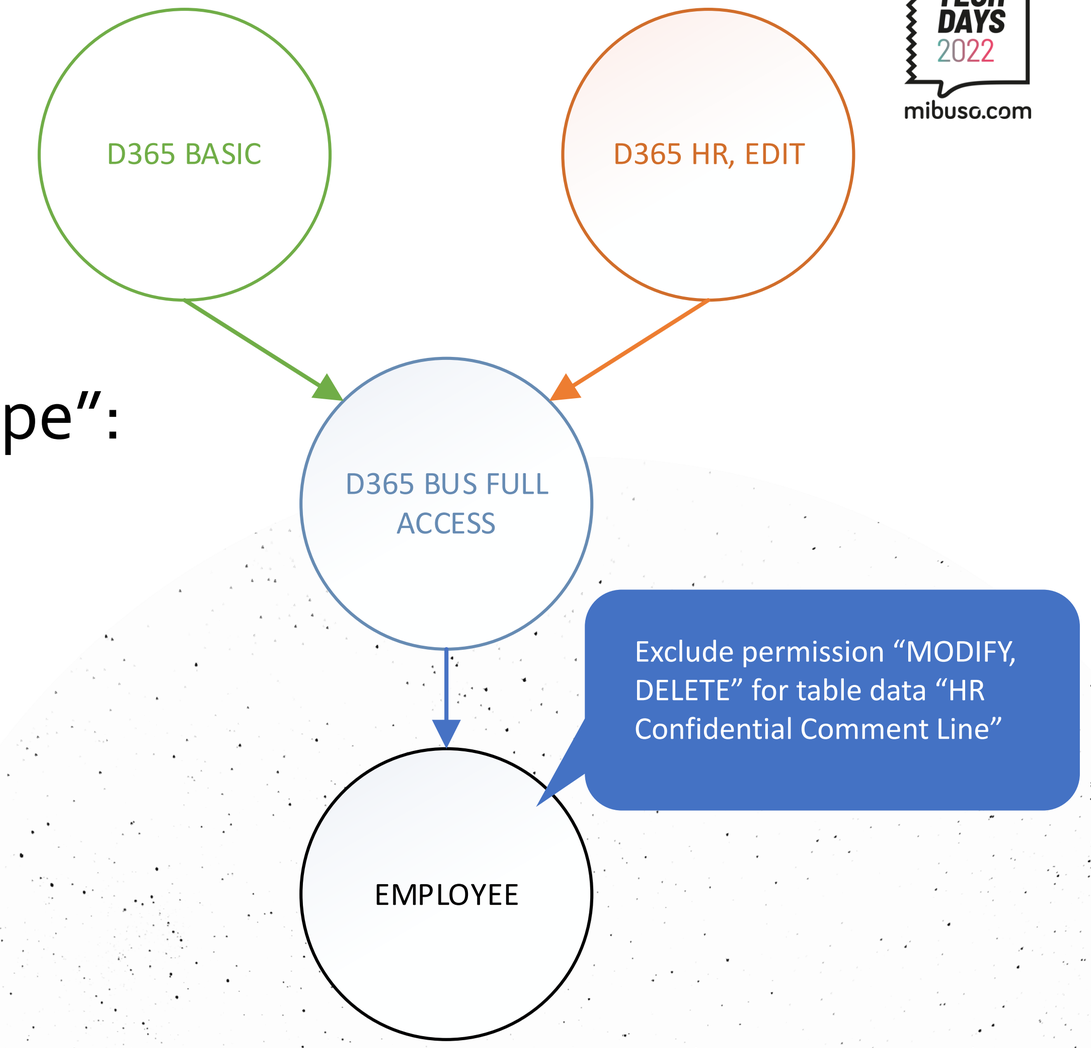


Exclude permissions

2022 W1: Exclude permission

"Tenant Permission" table has a new field "Type":

- "Include" (default)
- "Exclude"



"EMPLOYEE" permissions =

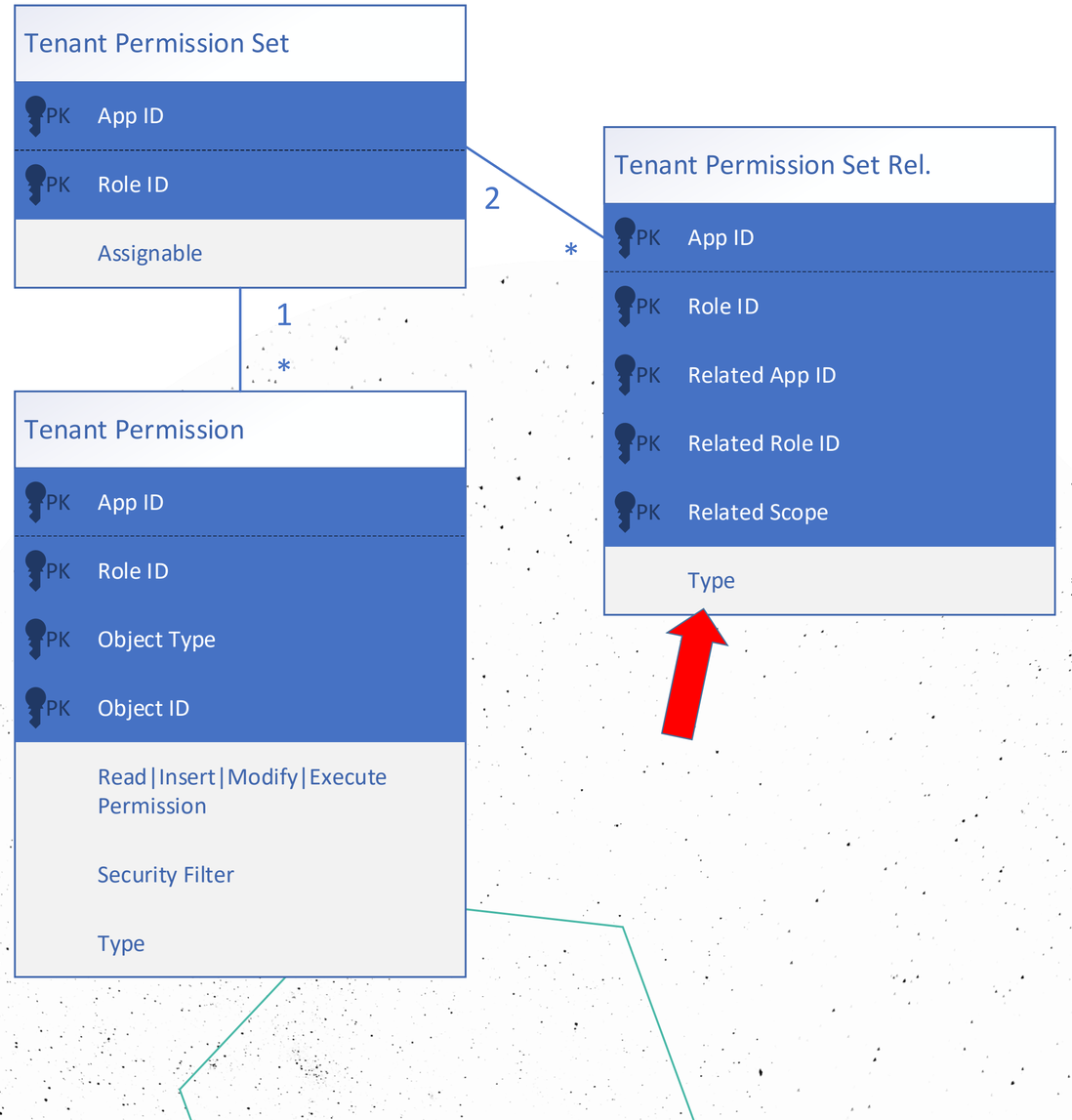
("D365 BUS FULL ACCESS" permissions +

Permissions) - "EMPLOYEE" exclude permissions

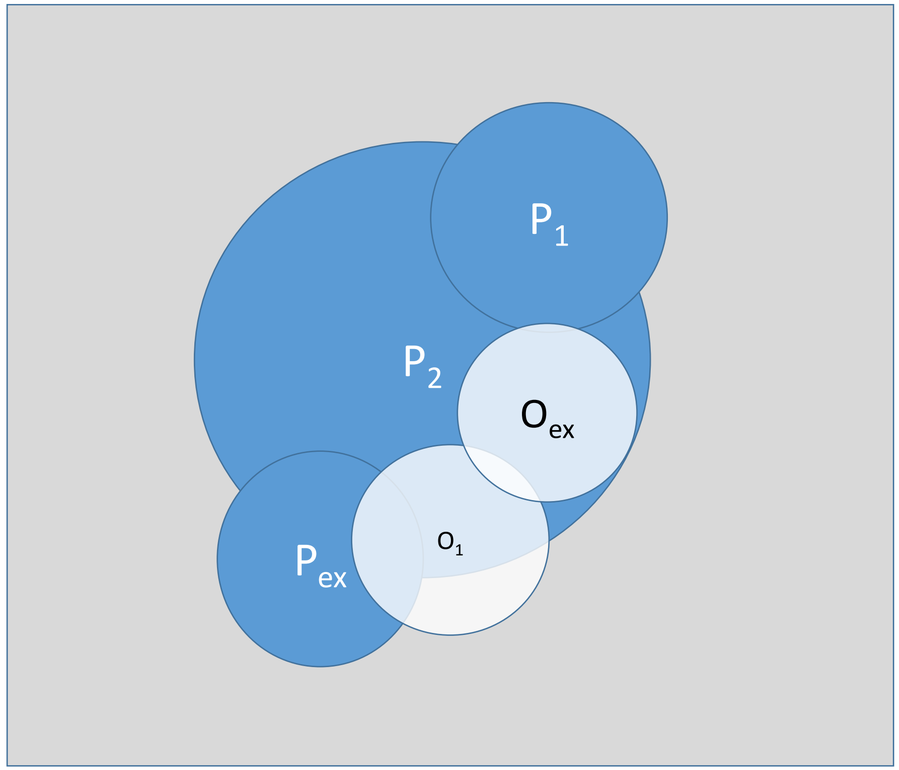
{ Which excludes "MODIFY, DELETE" for "HR Confidential Comment Line" }

Excluding permission sets

- Notice the Type field on relations
  - Include (default)
  - Exclude



How exclusion of permission sets work



Warning – Math notation

Permission set P_{top}
- Includes permission sets P_1 and P_2
- Includes explicit permissions P_{ex}
- Excludes permission set O_1
- Excludes explicit permissions O_{ex}

$$P_{top} = (P_1 \cup P_2 \cup Pe_x) / (O_1 \cup Oe_x)$$

∪: Union

/:Complement

Blue area are your permissions

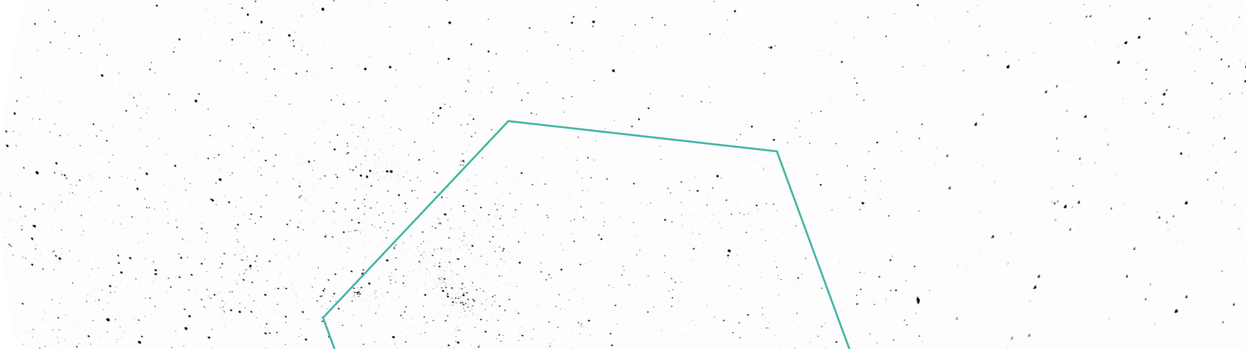
Excluding permissions explained

Mental model

Calculate the expanded list of included permissions
Calculate the expanded list of excluded permissions
‘Subtract’ the excluded permissions from the included

Excluding permission sets

Same as excluding the contained permissions



Exclude permission set: Changes

BC/NAV TECH DAYS 2022
mibuso.com

BC 2022 Wave 1

- Excluded permission sets were 'skipped
  - Required exact name match
  - No effect if permission set was not present
  - Depended on internals of permission set

BC 2022 Wave 2

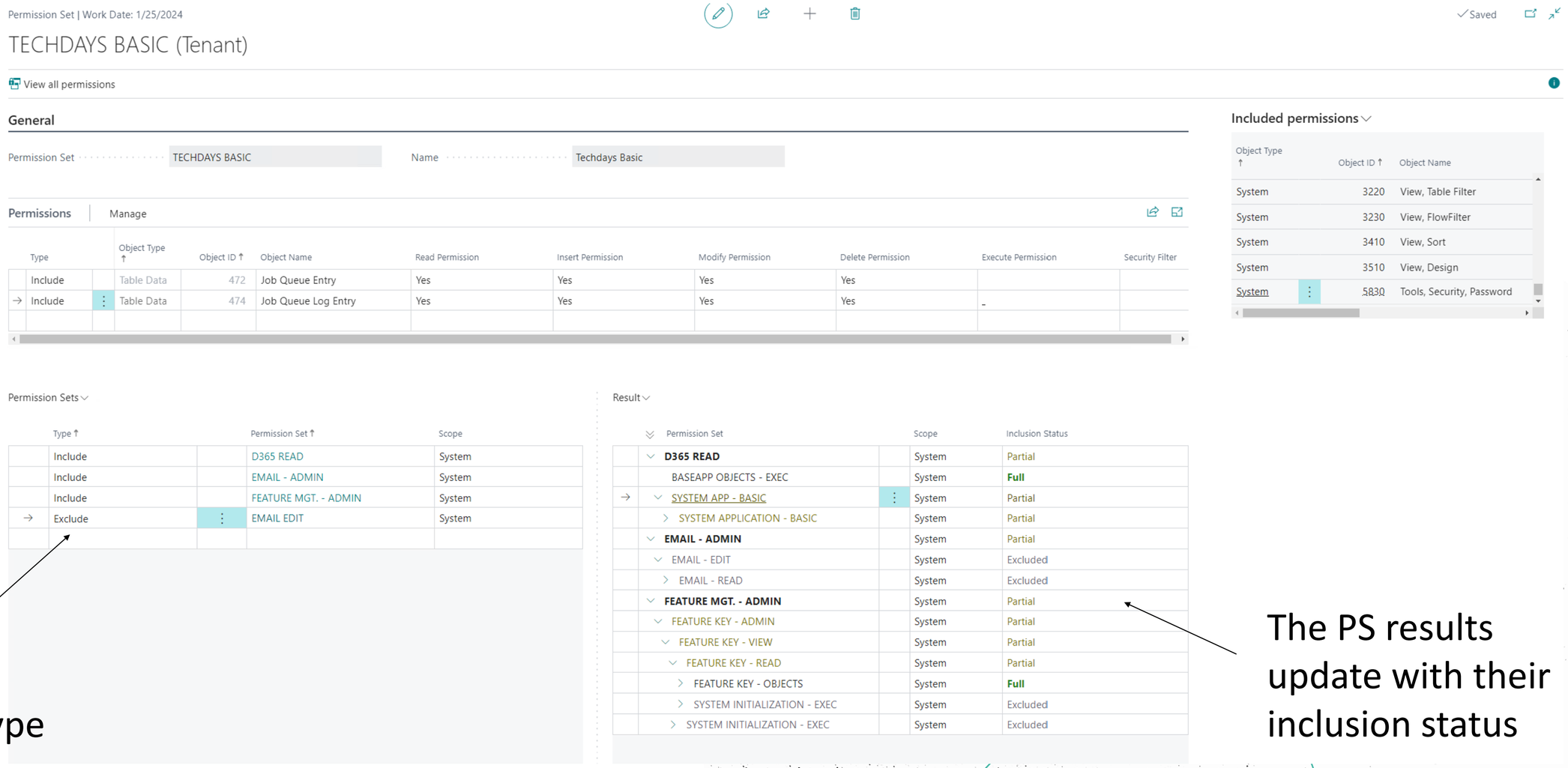
- Excluded permission sets are calculated – and then subtracted
  - Same behavior regardless of internals

Excluding permission set in metadata

In extensions you can only exclude permission *sets*

```
permissionset 132218 "Test Tables - Restricted"
{
    Assignable = true;

    IncludedPermissionSets = "System Application Test Tables",
                             "Local Test Tables";

    ExcludedPermissionSets = "Restricted Local Test Tables"

    Permissions = tabledata "All-Keys Type" = RIMD,
                  tabledata "Amount Auto Format Test Table" = RIMD,
```

Excluding permissions in the UI

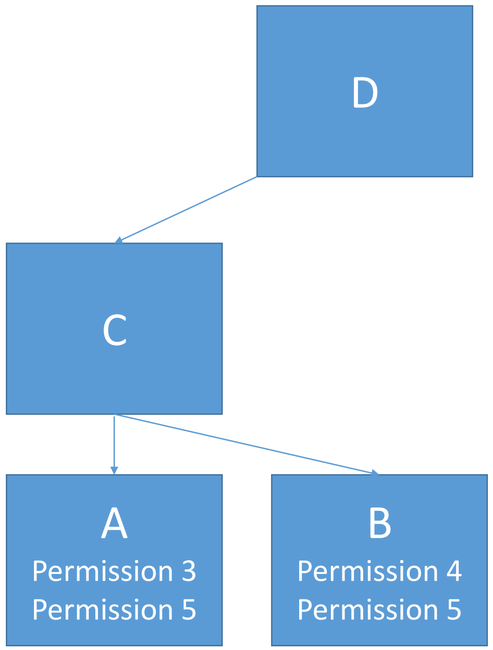


Exclude by updating the type

The PS results update with their inclusion status

Examples

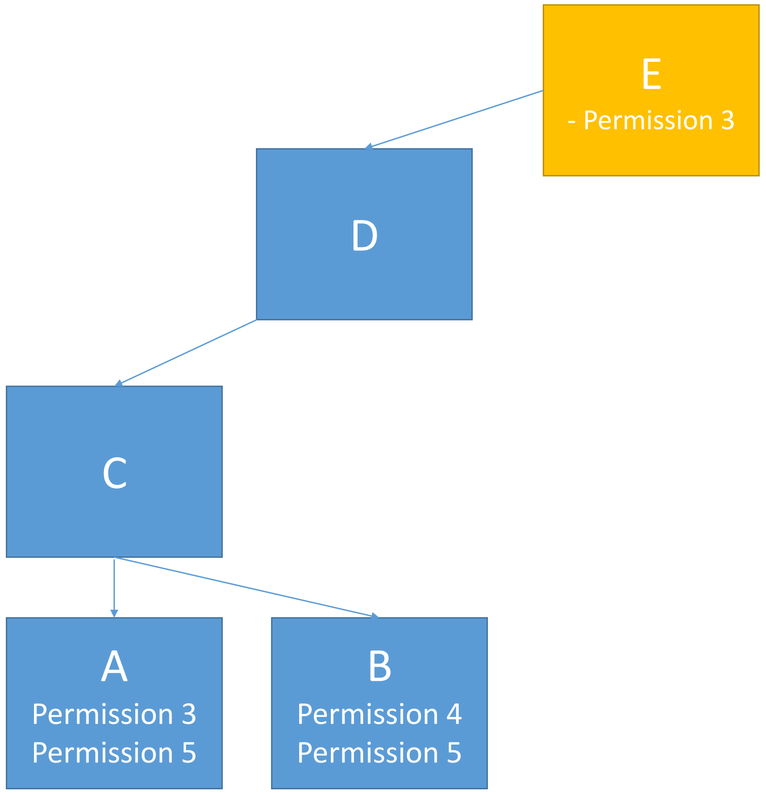
Examples – Base case



Expanded permissions:

- Permission 3
- Permission 4
- Permission 5

Excluding individual permissions - 1



E includes D and excludes P 3

What do you expect?

Permission 3, 4, 5

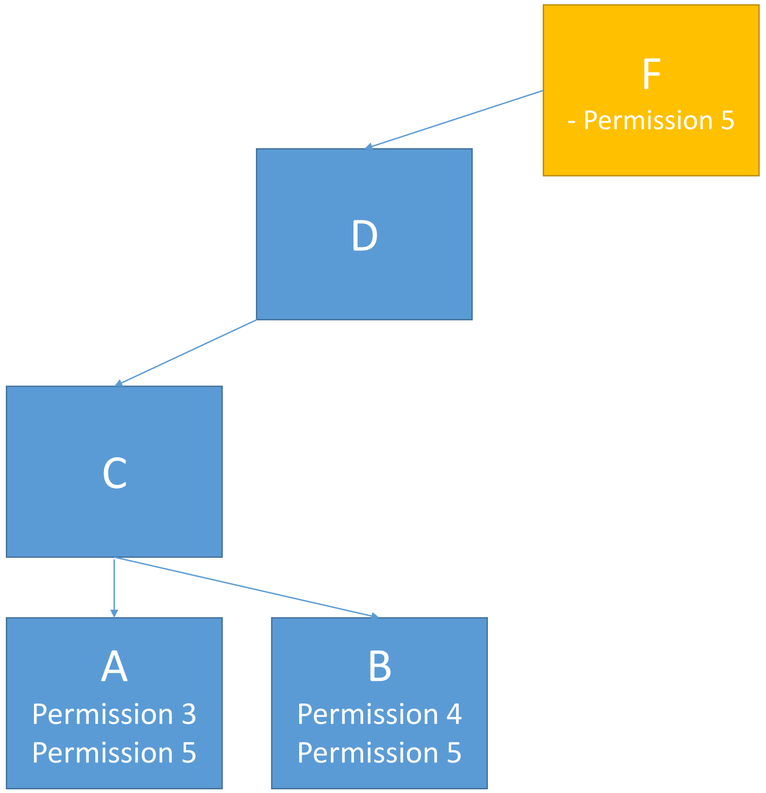
Permission 4, 5

Expanded permissions

Permission 4

Permission 5

Excluding individual permissions - 2



F includes D and excludes P 5

What do you expect?

Permission 3, 4, 5

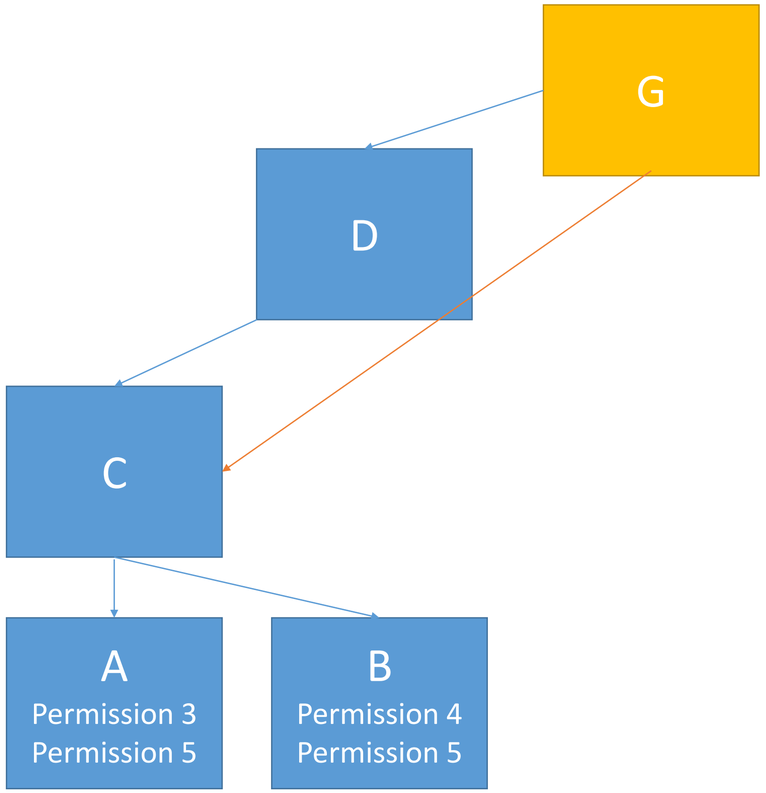
Permission 3, 4

Expanded permissions

Permission 3

Permission 4

Excluding full permission set - 1



G includes D and excludes C

What do you expect?

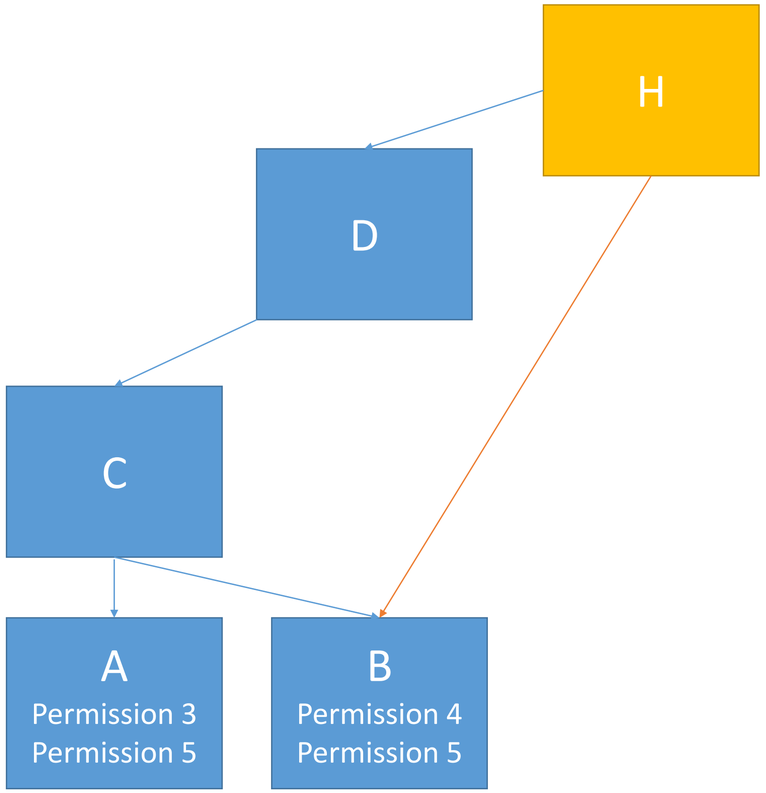
<Empty>

Permission 3, 4, 5

Expanded permissions

<Empty>

Excluding full permission set – 2



H includes D and excludes B

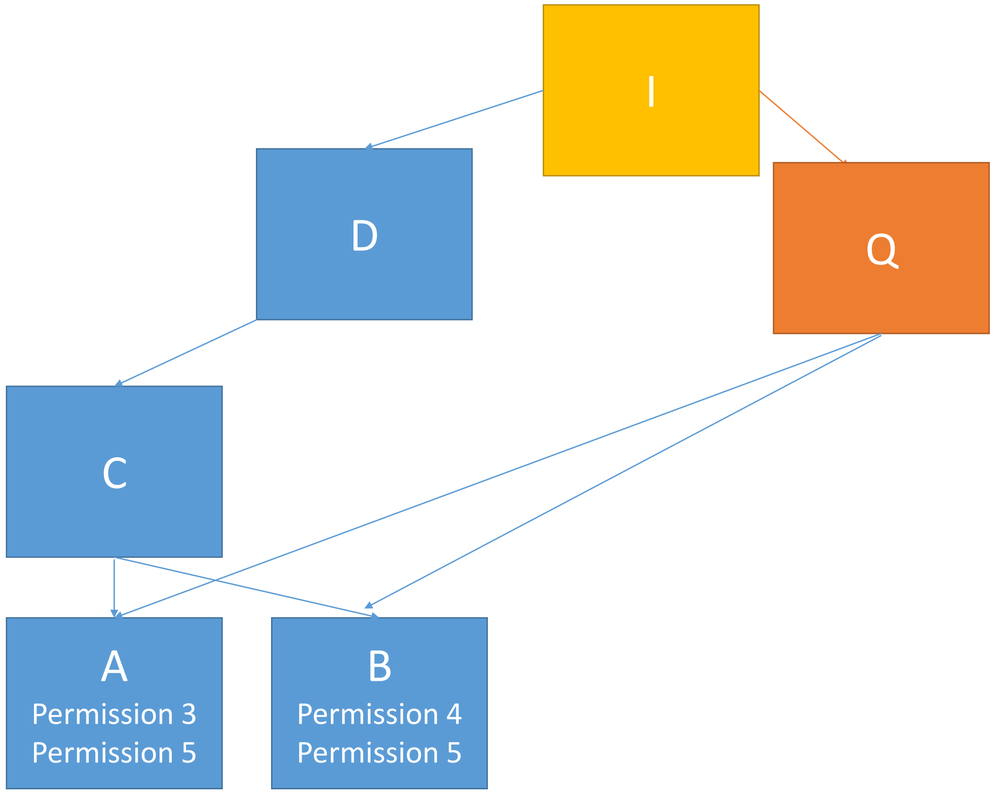
What do you expect?

- <Empty>
- Permission 3
- Permission 3, 5

Expanded permissions

- Permission 3

Excluding full permission set – new set



I includes D and excludes Q
(Q includes A and B)

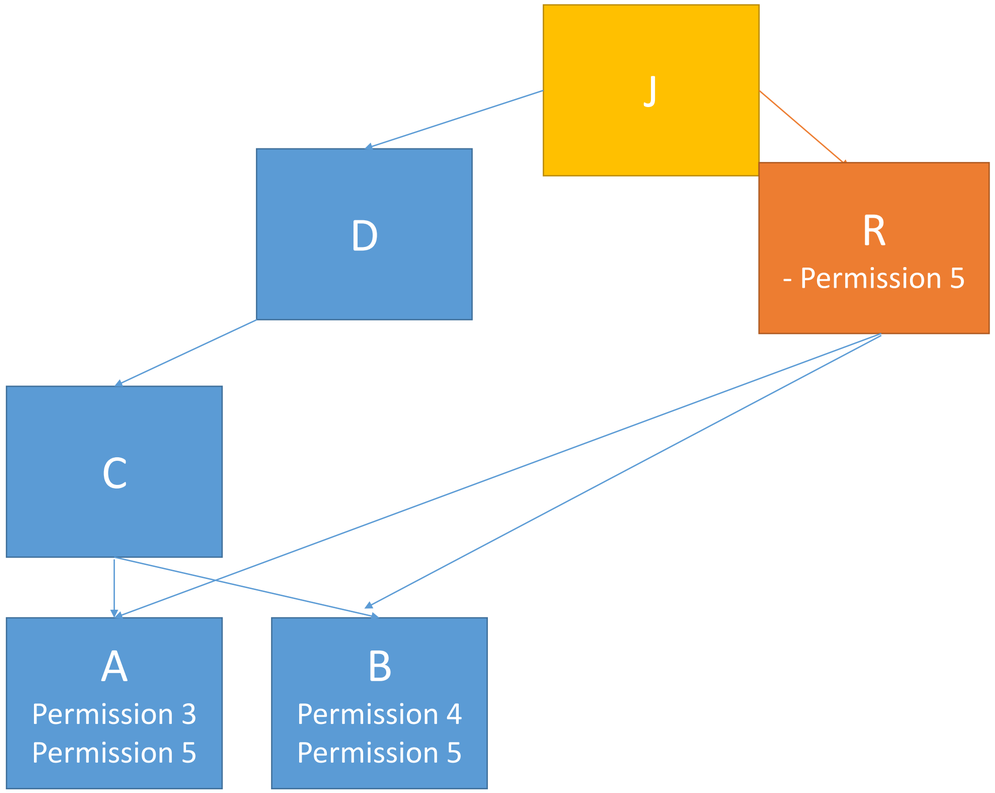
What do you expect?

- <Empty>
- Permission 3, 4, 5
- <Error>

Expanded permissions

- <Empty>

Excluding full permission set – new set



J includes D and excludes R

(R includes A, B and excludes P 5)

What do you expect?

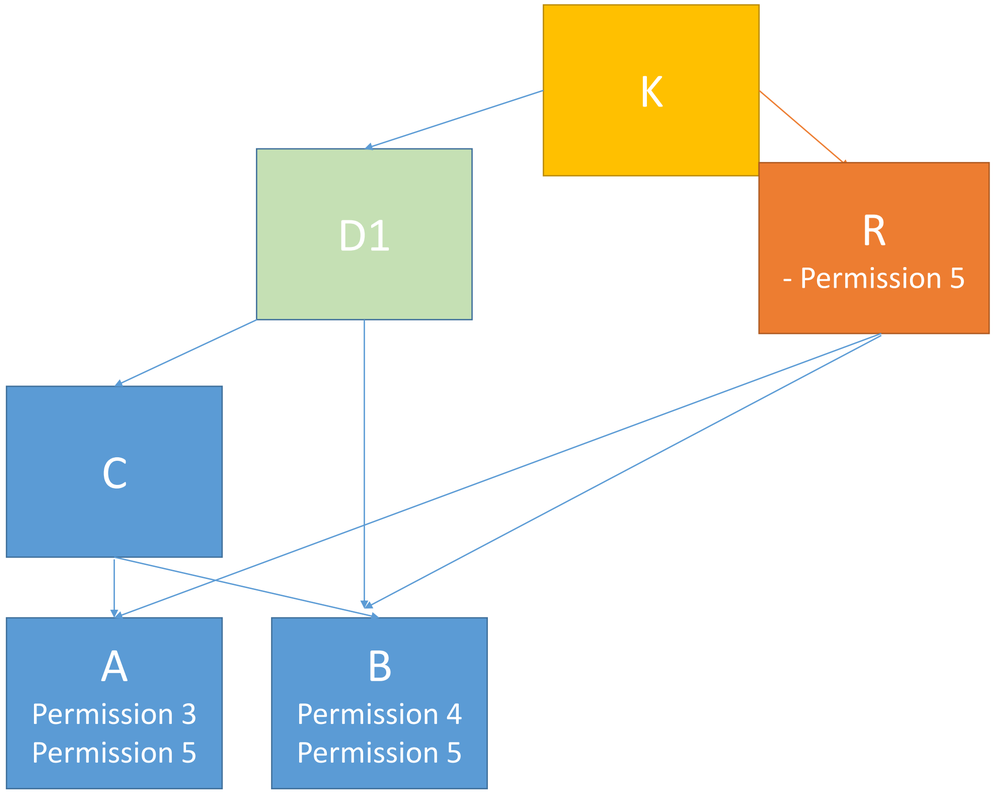
<Empty>

Permission 5

Expanded permissions

Permission 5

Excluding full permission set – new set



K includes D1 and excludes R

(R includes A, B and excludes P 5)

What do you expect?

<Empty>

Permission 4, 5

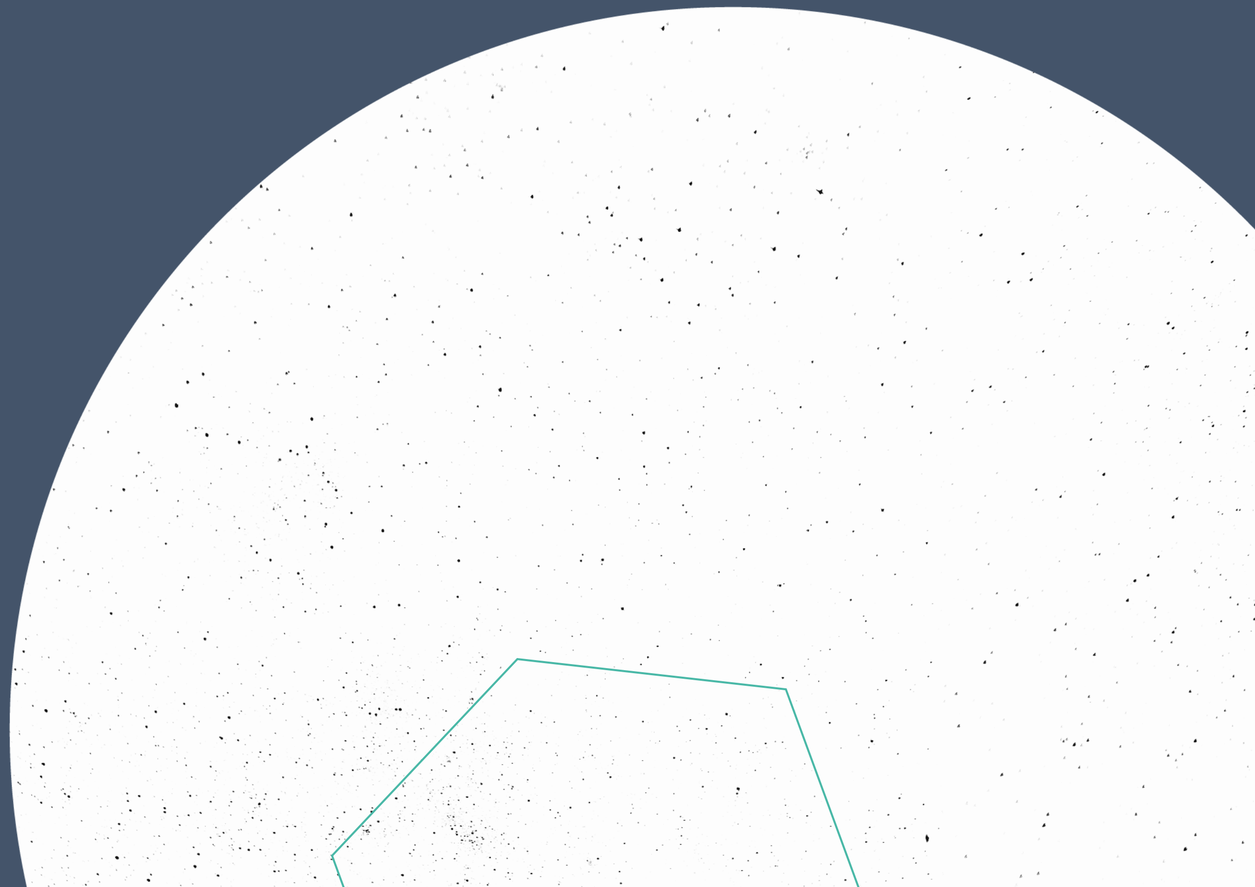
Permission 5

Expanded permissions

Permission 5

Demo

Composable Permission Sets of above examples



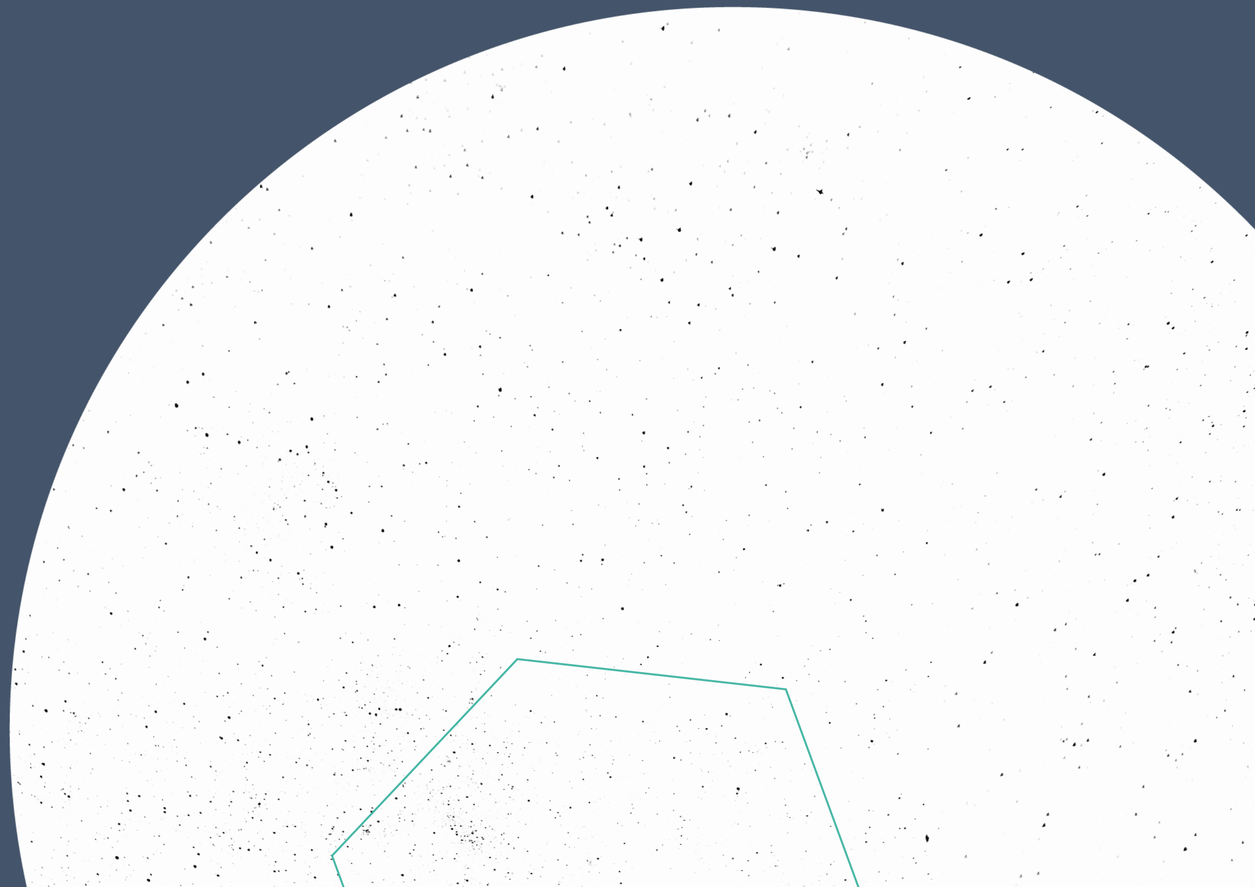
Excluding permissions - Considerations

- ‘Surgical’ exclusion of capabilities
  - Excluding existing permission sets will often exclude too much
  - Create a small permission set with the ones you want to exclude
- Permission set owner should rather include permissions
- Works best with small permission sets

Copy Permission Set

Demo

Copying a permission set



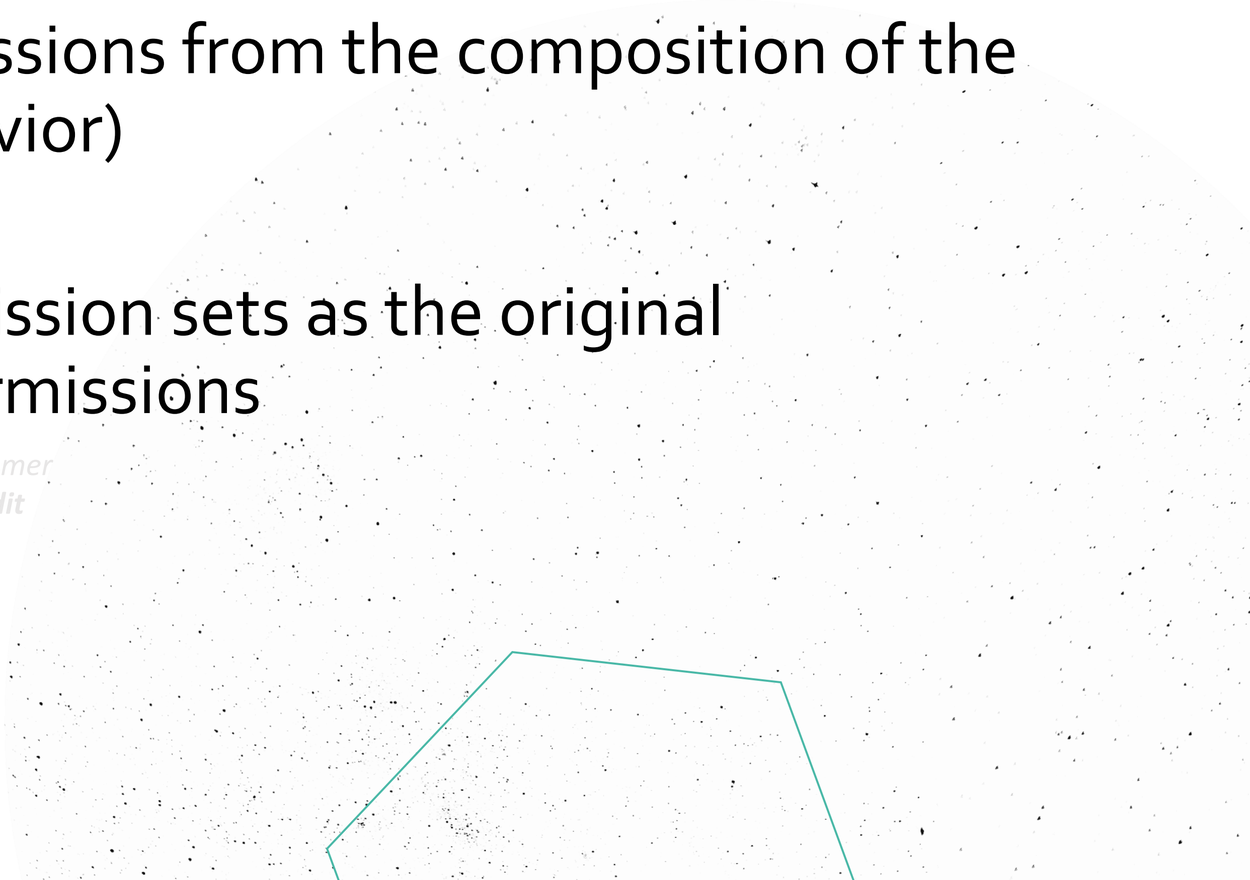
Different ways of copying a permission set

BC/NAV TECH DAYS 2022
mibuso.com

Reference: New set includes the original permission set (*default*)

Flat List: New set includes all the permissions from the composition of the original permission set (“old” copy behavior)

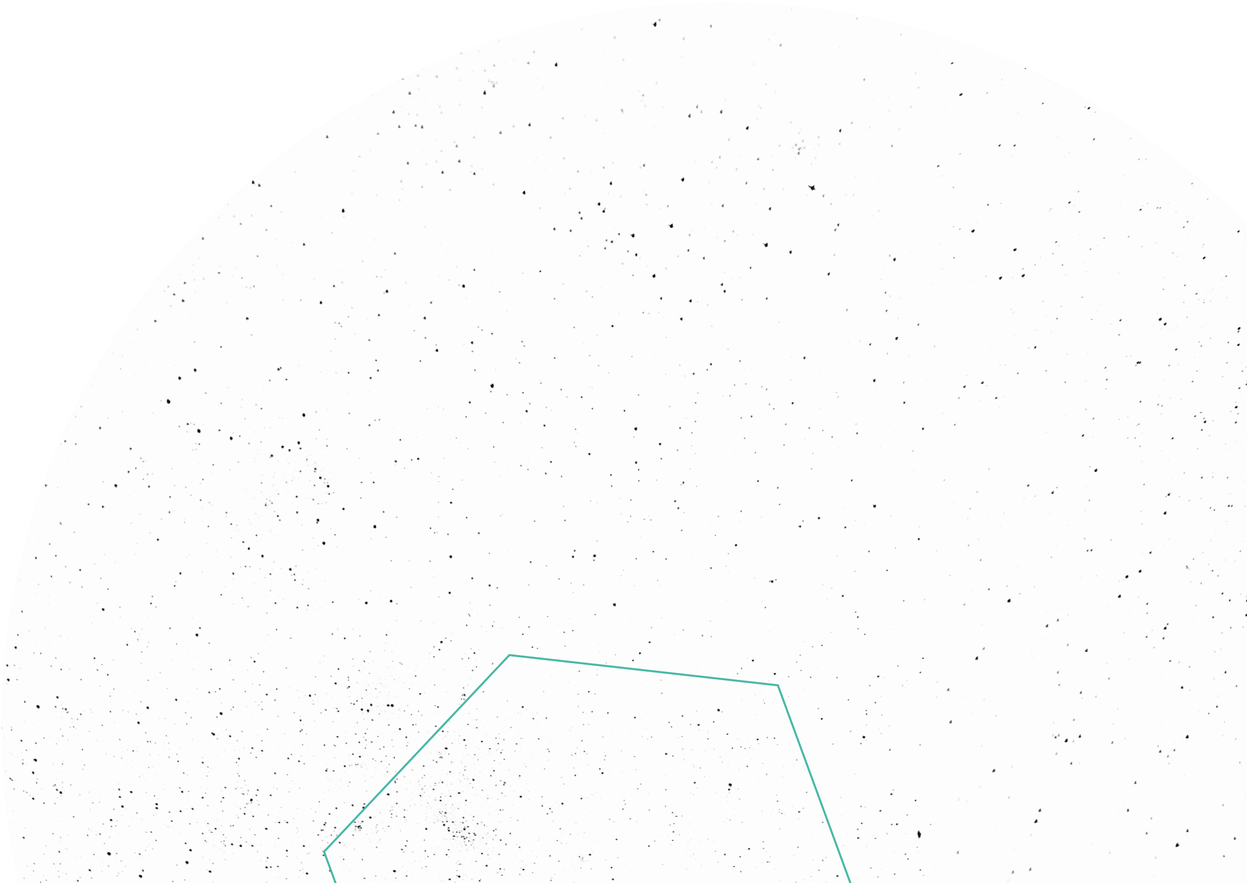
Clone: New set includes the same permission sets as the original permission set and defines the same permissions



Exported XML Permission Set - Old

BC/NAV TECH DAYS 2022
mibuso.com

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<PermissionSets>
 <PermissionSet AppID="{437DBF0E-84FF-417A-965D-ED2BB9650972}" RoleID="D365 FULL ACCESS" RoleName="Dynamics 365 Full access" Scope="System">
  <Permission>
   <ObjectType>Table Data</ObjectType>
   <ObjectID>3</ObjectID>
   <ReadPermission>Yes</ReadPermission>
   <InsertPermission>Yes</InsertPermission>
   <ModifyPermission>Yes</ModifyPermission>
   <DeletePermission>Yes</DeletePermission>
  </Permission>
 </PermissionSet>
</PermissionSets>
```

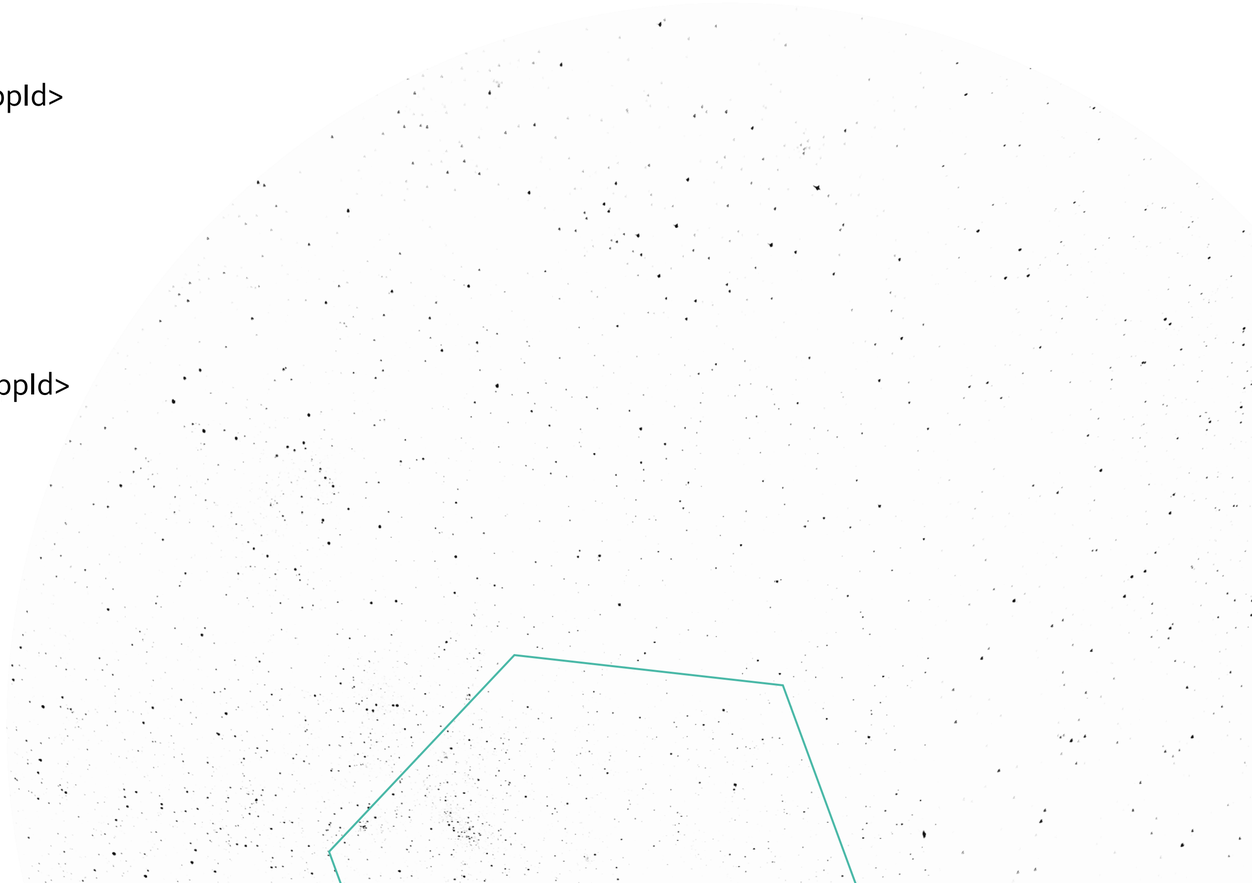


Exported XML System Permission Set - New

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<PermissionSets Version="2.0">
 <PermissionSet AppID="{437DBF0E-84FF-417A-965D-ED2BB9650972}" RoleID="D365 BASIC Copy" RoleName="Dynamics 365 Basic access copy" Assignable="Yes">
  <PermissionSetRel>
   <ObjectType>Include</ObjectType>
   <ObjectRelatedRoleId>LOGIN</ObjectRelatedRoleId>
   <ObjectRelatedAppId>{63CA2FA4-4F03-4F2B-A480-172FEF340D3F}</ObjectRelatedAppId>
  </PermissionSetRel>
  <PermissionSetRel>
   <ObjectType>Include</ObjectType>
   <ObjectRelatedRoleId>SESSION - EDIT</ObjectRelatedRoleId>
   <ObjectRelatedAppId>{63CA2FA4-4F03-4F2B-A480-172FEF340D3F}</ObjectRelatedAppId>
  </PermissionSetRel>
  <Permission>
   <ObjectType>Table Data</ObjectType>
   <ObjectID>3</ObjectID>
   <ReadPermission>Yes</ReadPermission>
  </Permission>
 </PermissionSet>
</PermissionSets>
```

Exported XML Tenant Permission Set- New

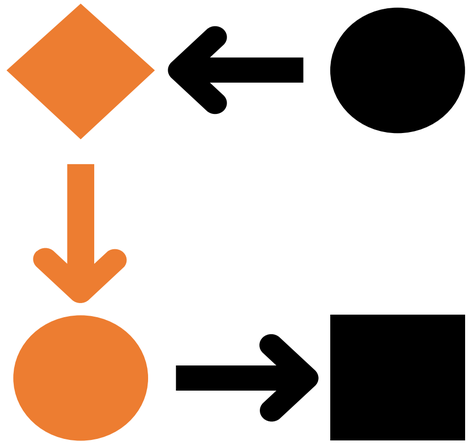

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<PermissionSets Version="2.0">
 <TenantPermissionSet AppID="{00000000-0000-0000-0000-000000000000}" RoleID="TEST2" RoleName="test3" Assignable="Yes">
  <TenantPermissionSetRel>
   <ObjectType>Exclude</ObjectType>
   <ObjectRelatedScope>System</ObjectRelatedScope>
   <ObjectRelatedRoleId>TELEMETRY - EXEC</ObjectRelatedRoleId>
   <ObjectRelatedAppId>{63CA2FA4-4F03-4F2B-A480-172FEF340D3F}</ObjectRelatedAppId>
  </TenantPermissionSetRel>
  <TenantPermissionSetRel>
   <ObjectType>Include</ObjectType>
   <ObjectRelatedScope>System</ObjectRelatedScope>
   <ObjectRelatedRoleId>BLOB STORAGE EXEC</ObjectRelatedRoleId>
   <ObjectRelatedAppId>{9856AE4F-D1A7-46EF-89BB-6EF056398228}</ObjectRelatedAppId>
  </TenantPermissionSetRel>
  <TenantPermission>
   <ObjectType>Table Data</ObjectType>
   <ObjectID>472</ObjectID>
   <ReadPermission>Yes</ReadPermission>
  </TenantPermission>
 </TenantPermissionSet>
</PermissionSets>
```

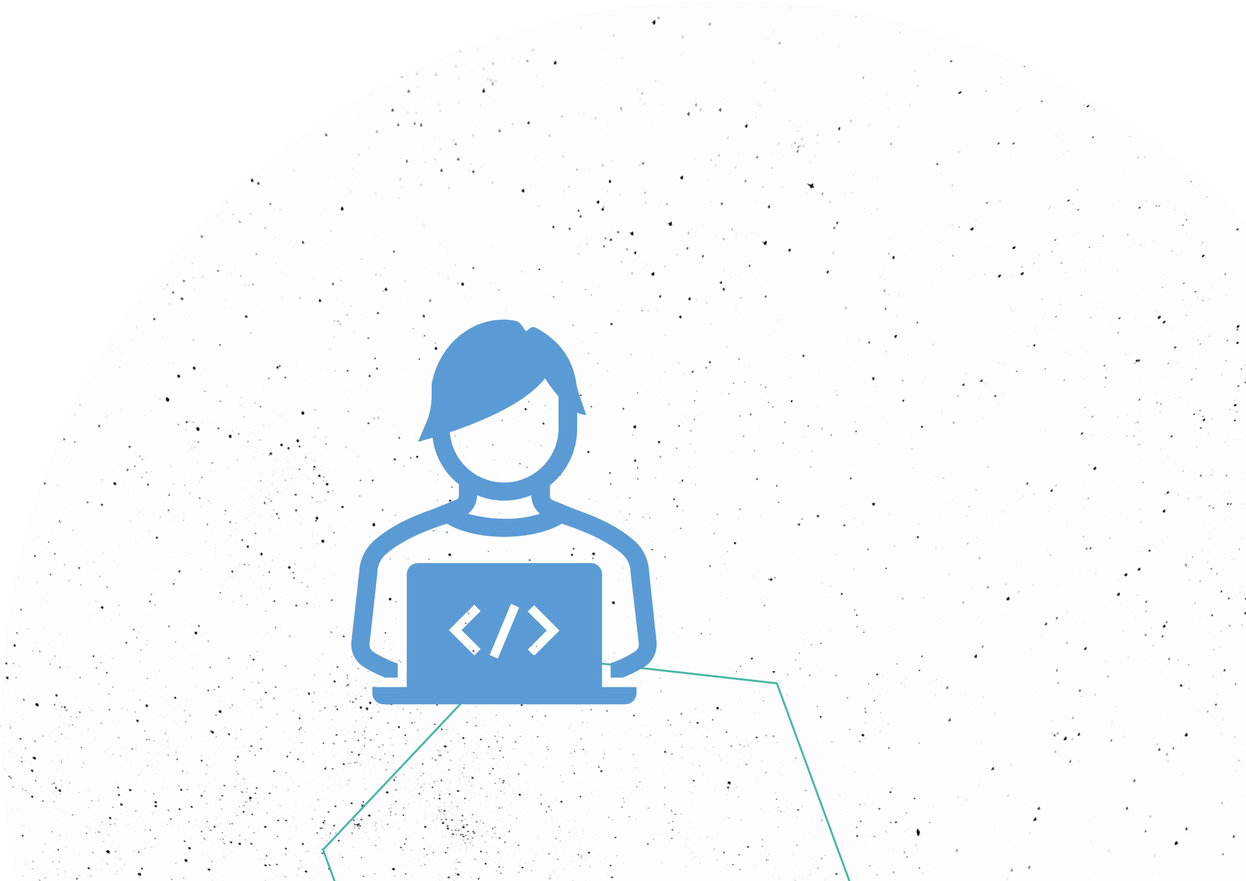


Inherent Permissions

Inherent Permissions

- **Ability to elevate user permissions in a given code context**
- **Only granted during the specific method execution**
- Puts power in the hands of the developer

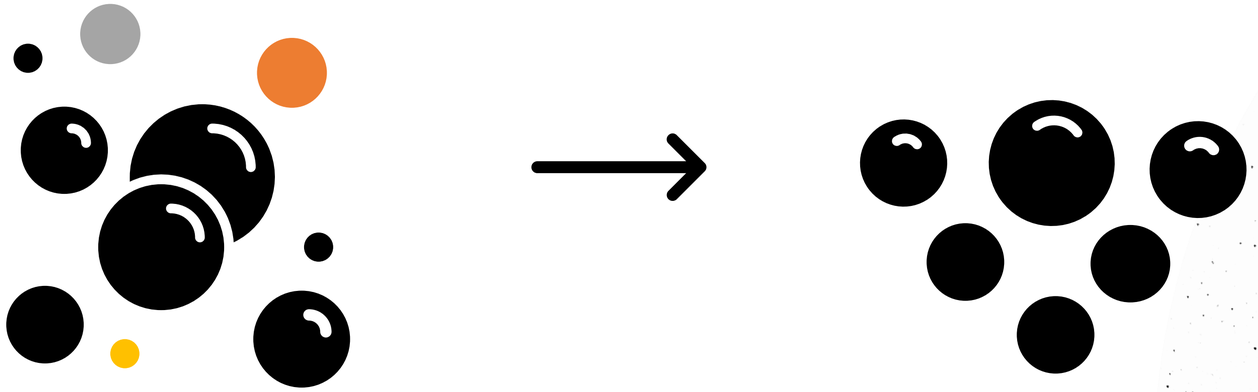




Inherent Permissions

- Should no longer be explicitly added to the permission set
- More clean and comprehensible permission sets
- Improve the stability of the critical code paths

- Can only be used for *your own objects*



Inherent Permissions - syntax

New method attribute

```
[InherentPermissions(PermissionObjectType, ObjectId, Permission)]
procedure MyProcedure()
```

Within this procedure, user has extra permissions and entitlements

You *can* limit to either permissions or entitlements

```
[InherentPermissions(PermissionObjectType, ObjectId, Permission, InherentPermissionsScope)]
procedure MyProcedure()
```

InherentPermissions code example

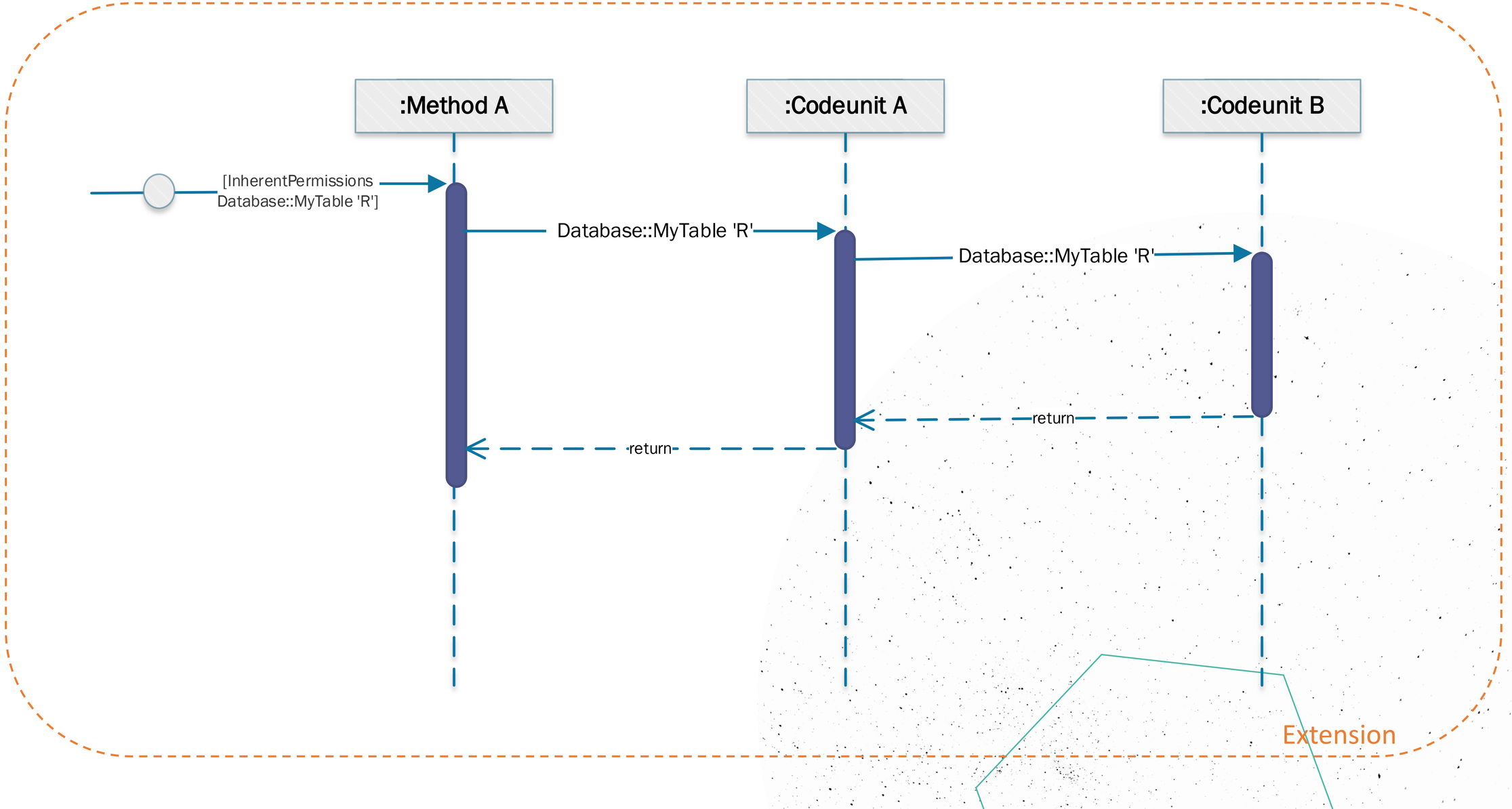
New AL enum PermissionObjectType

Object to be assigned InherentPermissions

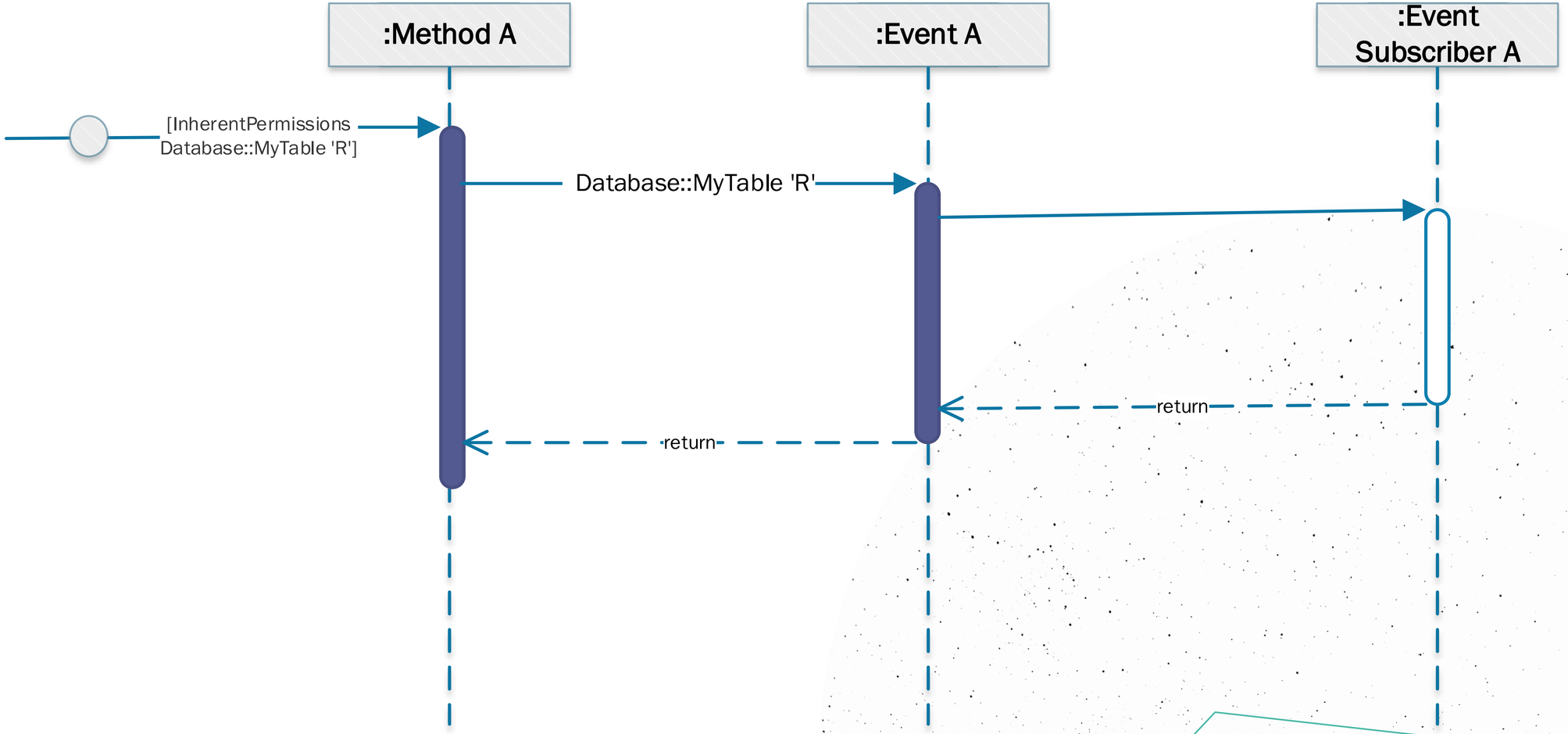
```
[InherentPermissions(PermissionObjectType::TableData, Database::"G/L Entry", 'r')]
8 references
procedure GetDefaultWorkDate(): Date
```

Within this method, all users have indirect permission (and entitlement) to read from G/L Entry table.

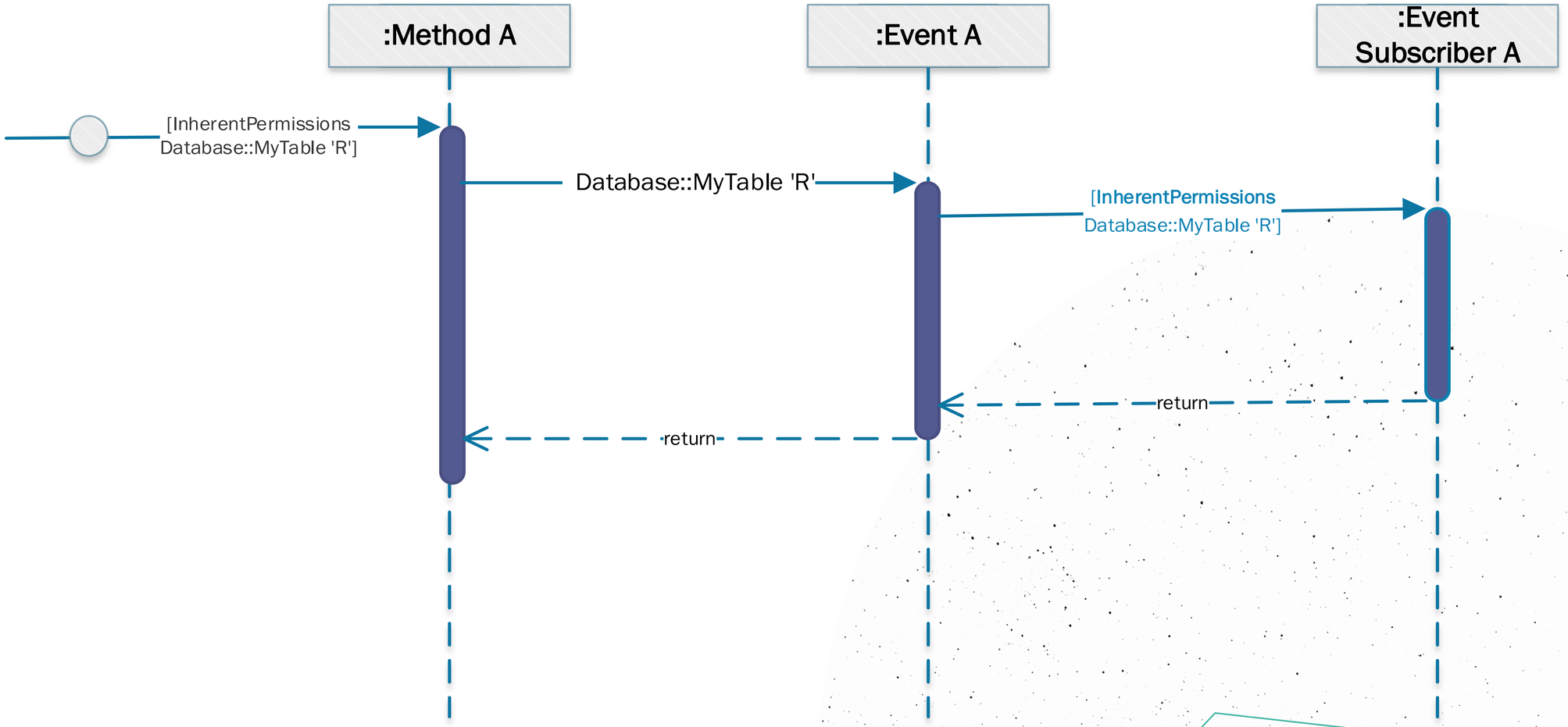
Scope follows down the stack



Event Subscribers



Event Subscribers



The attribute can be added explicitly to the subscriber

When and how to use Inherent permissions

Small dedicated procedures performing a task

- Avoid adding InherentPermissions at top level
- Ideally avoid calling other methods or raising events

Critical code paths

- Code that should run, regardless of the user

‘System tasks’ that do not expose data to users

- Initialization of default values
- Derived information like: Counts of records, is feature enabled

No need for Stan to restrict access

- Stan cannot remove privileges to run this code

Wrap Up

Summary

Well defined, understandable building blocks

- Hierarchical permission sets
- Better UI

Reduce maintenance burden

- Reference permission sets instead of copying

Simplify permission sets

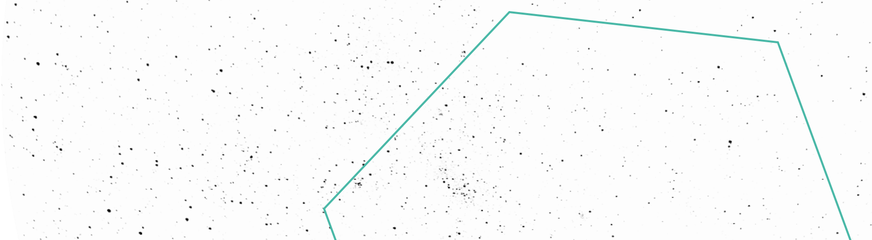
- InherentPermissions attribute

Greater flexibility

- Exclude permissions and permission sets

Next steps

- Further componentize permission sets
- InherentPermissions on objects
- Troubleshooting tools
- Improved auditing and telemetry



BC/NAV TECH DAYS 2022

mibuso.com

Q&A

10 YEAR ANNIVERSARY

www.bctechdays.com

BC/NAV TECH DAYS 2022

mibuso.com

Thank You!

10 YEAR ANNIVERSARY

www.bctechdays.com