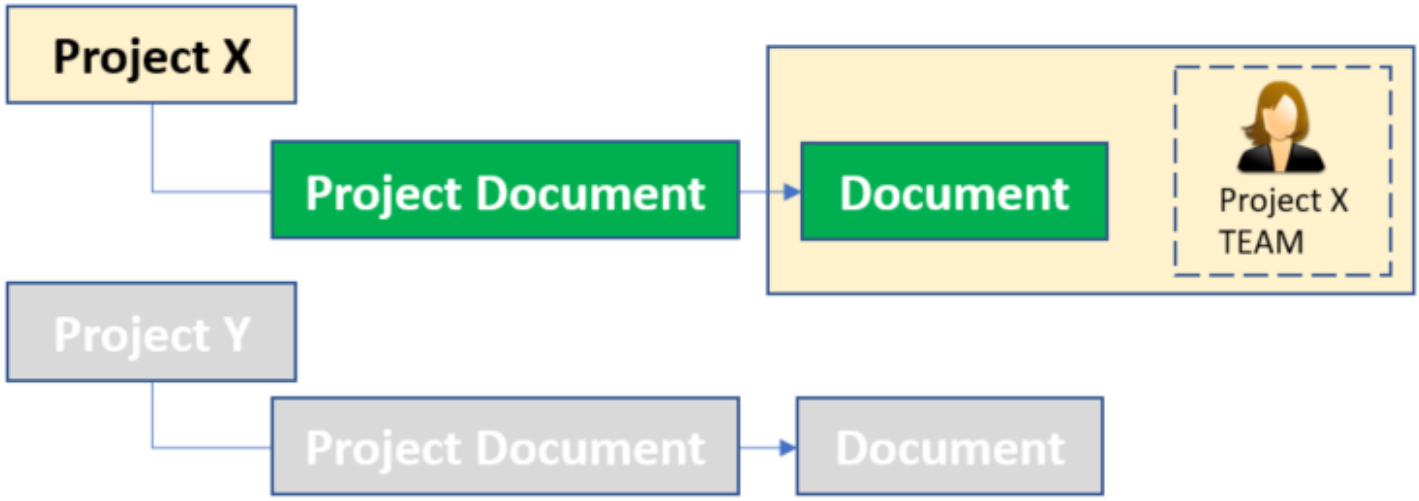
Overview

Domain Access Control (DAC) is an advanced form of access control that gives Administrators the ability to create relationship-based access controls within Aras Innovator. This type of access control can be used to grant access to specific data within the scope of a “project” or “workspace”.

DAC works in conjunction with standard Aras Innovator role-based permissions and MAC policies. In order to have access to an item, users must have permissions through either DAC or standard Aras Innovator role-based permissions. In addition, a user must satisfy any applicable MAC Policies.

Relationship Structures as Access Control Domains

DAC grants access to items based on an item’s occurrence in a relationship structure under a Root item. The Root of the structure is the Domain scope - **Project X** at right. Access to related **Document** item can be controlled by DAC for the Project Domain. DAC can grant or elevate access to the Items within its domain.



DAC uses a ‘derived’ representation of existing physical relationships without making any changes to the source items, related items or relationships themselves. Derived Relationships allow DAC to track changes and organize access rules against the corresponding business data.

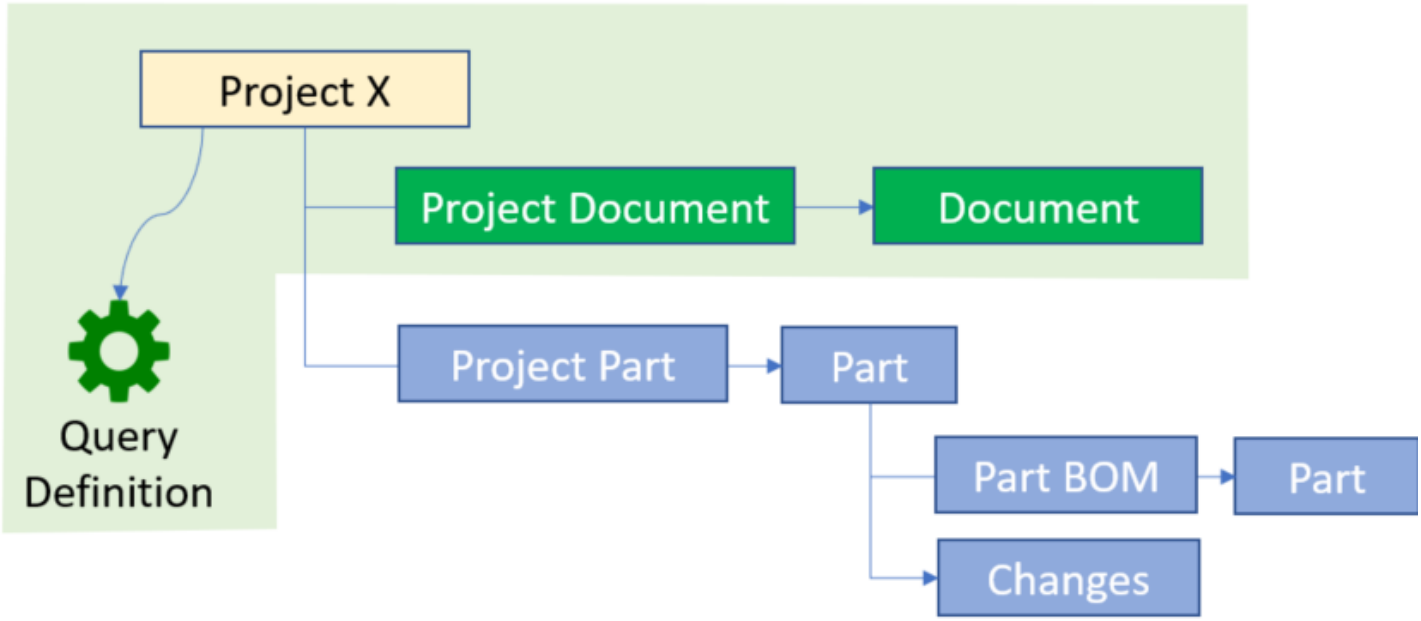
One or more branches or paths from a Root to a related item can be used to derive relationships. Each path and its endpoints define a DAC Subdomain. A DAC Domain contains all of the

Subdomains derived by one Query Definition against a Root item.

Query Definition

A standard Aras Query Definition is used to indicate which parts of an overall ItemType Relationship structure is to be used by DAC in an access control policy.

Specifically, a query is used to extract the relationship ‘paths’ from Root to Leaf items that will implement DAC access control. The Aras Query Builder is used to define a Query Definition for DAC.



Query Conditions, where-used references, and other Query Builder features are supported, thus providing a powerful method of identifying the specific paths, or ‘Subdomains’ within the Relationship structure where Access Rules will be set by DAC.

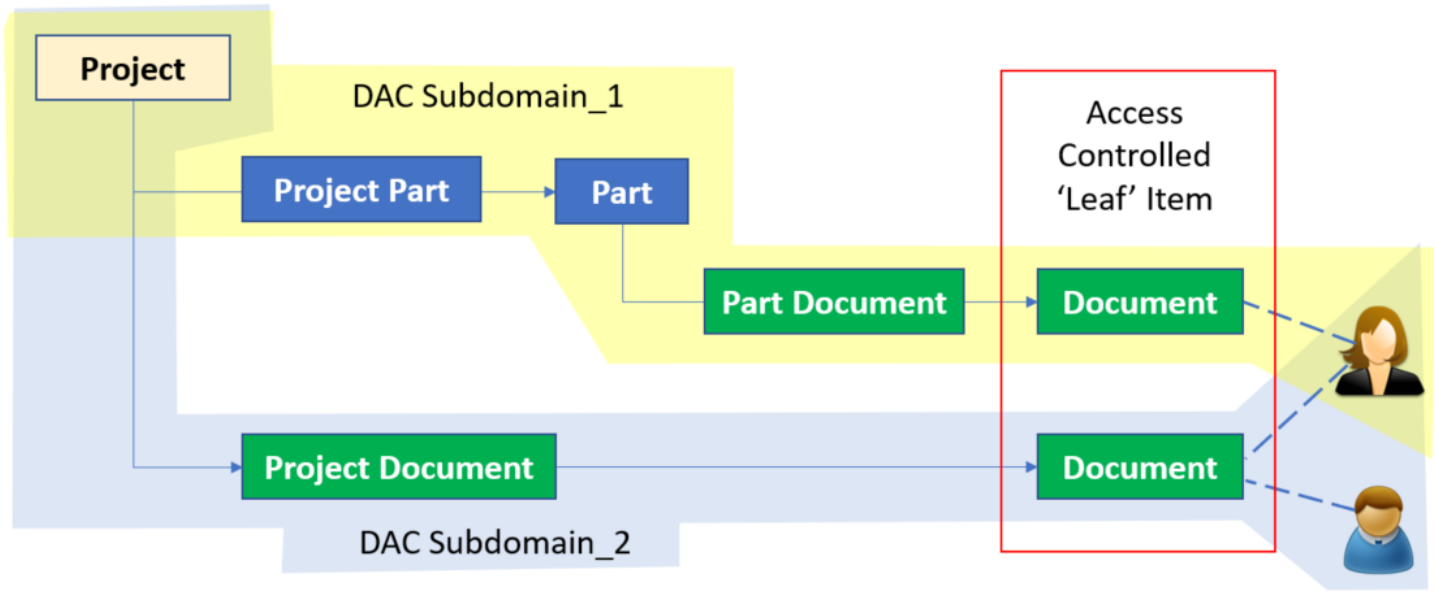
DAC Subdomains

Each branch or path within a Relationship structure can be used to implement a discreet set of Access Rules in a DAC **Subdomain**. There must be at least one Subdomain in any DAC definition.

In the following example **Project** is the root of the Domain and two subdomains are configured under DAC access control – *Subdomain_1* and *Subdomain_2.* Both subdomains control the **Document** ItemType in this case, but any relationship can be used. *Subdomain_1* has an intermediate **Part**

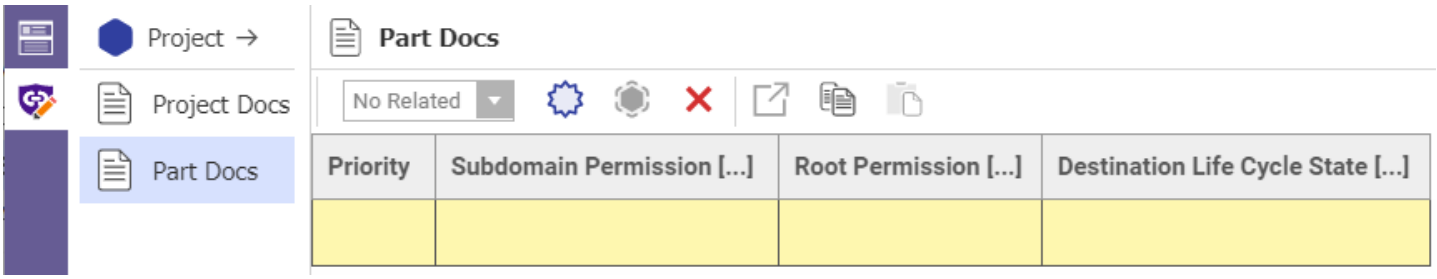
node and enforces different access rules than *Subdomain_2* for the same Document ItemType which is related directly to the Project.

The Document under Part can gain an elevated access for Engineering Roles via DAC, whereas DAC can grant 'Get' access to all members of the Project Team to Documents related to the Project directly, depending on business requirements.



DAC Subdomain Rules

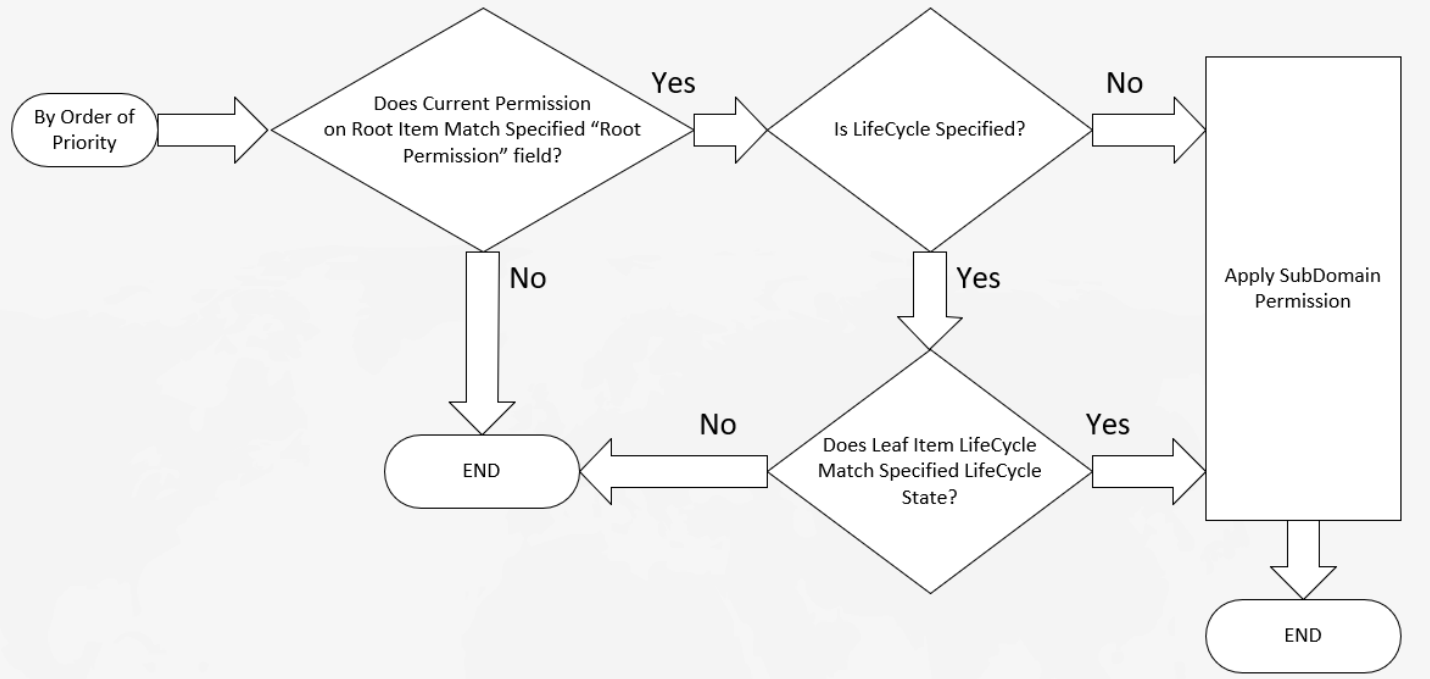
For each subdomain used in a DAC Definition, the administrator must define one or more Subdomain Rules to ensure that the correct permissions are granted to the Leaf item. A subdomain rule determines the proper permissions to grant based on input of the fields shown in Figure 4 and described in the table below.



Fields for granting permissions

Field	Description	Required?
Priority	Used to set the order in which the Rules are evaluated. This is useful when the same Subdomain Permission and Root Permission pair appear in more than one row, but with a different (or blank) Life Cycle State (see detail below).	Yes
Subdomain Permission	Permission to grant to Leaf Item if Rule is satisfied.	Yes
Root Permission	Permission to match with the current permission on the Root Item (e.g. Project).	Yes
Destination Life Cycle State	If specified it forces a match with the current Leaf Item lifecycle state.	Optional

When the DAC Policy is active the fields are evaluated at runtime as shown in Figure 5.



For example, consider subdomain rules described in the table below.

Subdomain rules

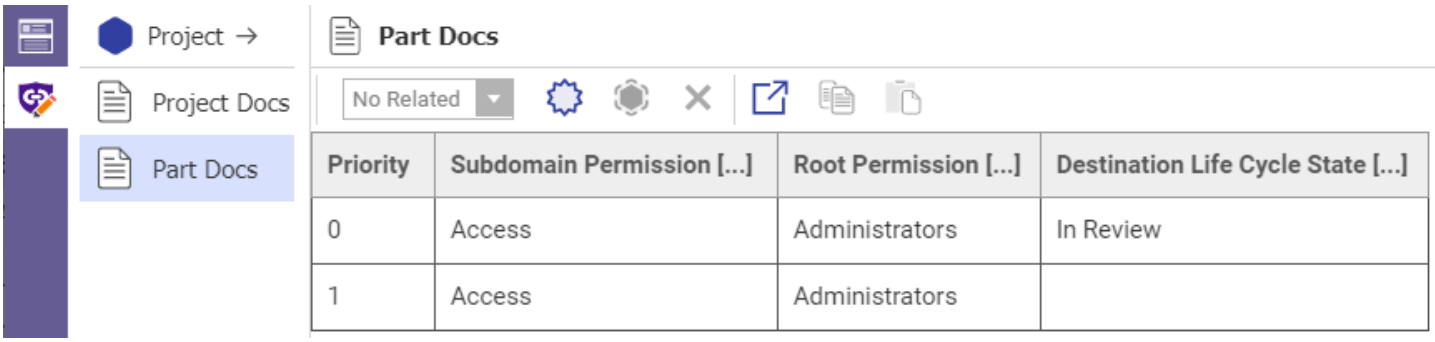
Subdomain Permission	Root Permission	Destination Life Cycle State
Preliminary_Document	Preliminary_Project	
ReadOnly_Document	Released_Project	Released
DiscoverOnly_Document	Released_Project	Superseded

If these Rules were set on the **Project->Document** subdomain it would imply the following:

- **If the Permission_ID on the Root (Project) is the ID of 'Preliminary_Project'** then apply 'Preliminary_Document' permission to the Leaf (Document).
- **If the Permission_ID on the Root (Project) is the ID of 'Released_Project' AND the LifeCycle State of the Leaf (Document) is 'Released'** then apply 'ReadOnly_Document' permission to the Leaf (Document).
- **If the Permission_ID on the Root (Project) is the ID of 'Released_Project' AND the LifeCycle State of the Leaf (Document) is 'Superseded** then apply 'DiscoverOnly_Document' permission to the Leaf (Document).

The **Priority** Field works as follows:

- No two rules may have the same Root Permission and LifeCycle State. This will issue a validation error when saving DAC Definition.
- If the Life Cycle State is left blank (implying 'any') then the Priority value must be a higher positive integer than any specified Life Cycle State.



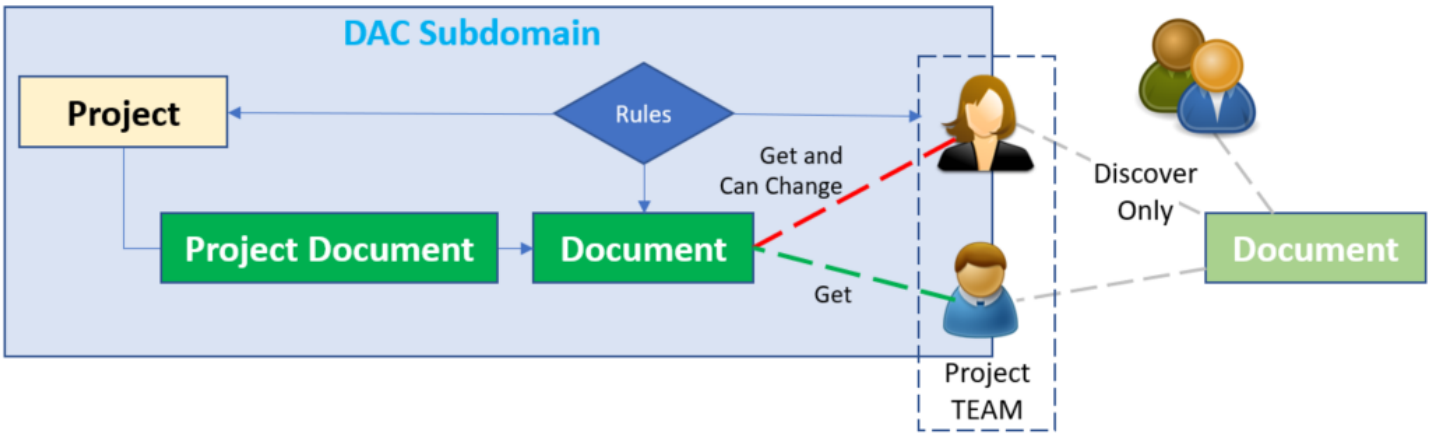
Priority	Subdomain Permission [...]	Root Permission [...]	Destination Life Cycle State [...]
0	Access	Administrators	In Review
1	Access	Administrators	

Example Scenario: Using a Subdomain Rule for Project Domain Access Control

The following is an example scenario of how a Subdomain Rule might be used:

- The Document Itemtype (outside of DAC) is configured with default "Discovery Only" access for All Employees.
  - Without DAC, MAC or Lifecycle driven permissions, this is the only access granted to employees.
  - Non-employees would have no access.
- The **Project Itemtype** has a **RelationshipType** 'Project Document' with a related **Document ItemType**.

    - In DAC, a Query Definition is used to identify this relationship as a DAC Subdomain.
- The **Access Policy** has the following business requirements:
    - All Project Team members receive “Get” access to Documents when added to the Domain and if the Root Permissions allow it. Access is then elevated for the Team. Users with the Administrator Role in the Project Team also getting “Can Change Access” rights in this case.
- A **Rule** is defined as follows:
    - **Root Permission** = ‘Released Project Permission.’
    - **Destination Life Cycle State** (blank).
    - **Subdomain Permission** = ‘Project Domain Member Access’:
    - **Team Member**: Get access.
    - Team Administrator: Get + Can Change Access.
- This Rule is added to the **Subdomain** and the **DAC Policy** is activated.
- A Document is added to the Project Relationship structure (Project Domain):
    - Result: Team members can now open the form and view the Document.
- Team members with the Administrator Role also have ‘Can Change’ access.



Subsequently rules can be added for different Root Permissions or LifeCycle States on the destination (Leaf) item, additional Team members, etc.

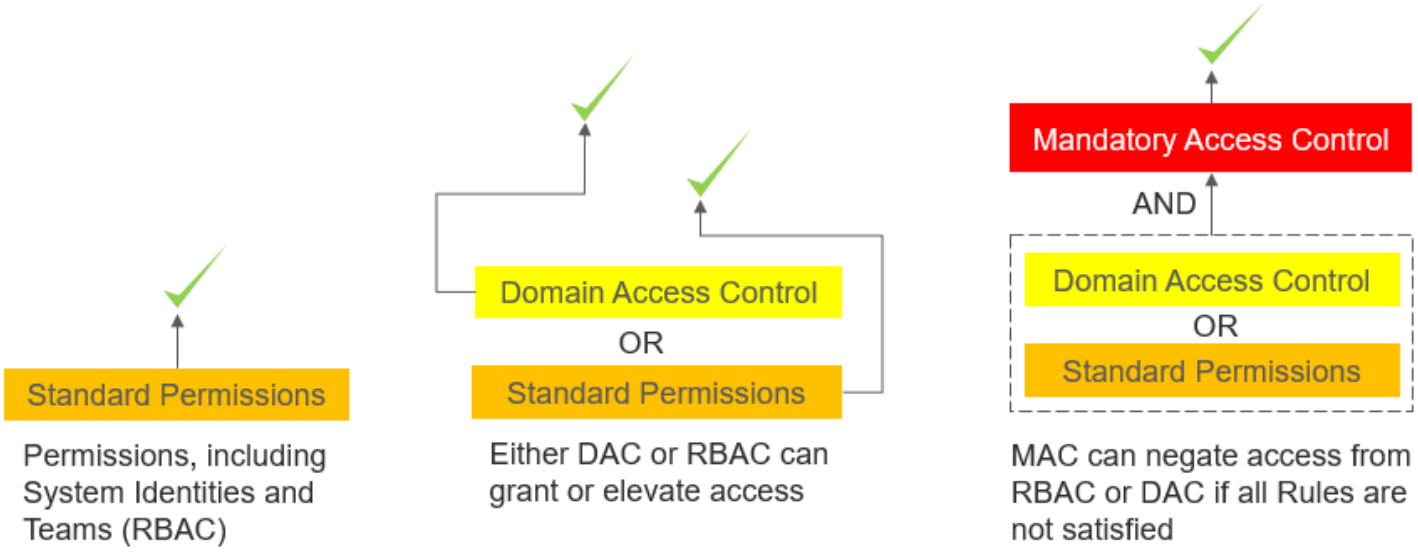
Mapping Team Roles from the Root Item Team

If there is a Team assigned to the Root item, and Roles are used in the Subdomain Permission (the permission that is assigned by the Rule calculation) then the Roles must match, otherwise there will not be a dynamic assignment by Role when the permission is applied. See Creating Subdomain Permissions for more information.

Domain Access Control (DAC) vs. RBAC and MAC

DAC can either grant or elevate access levels for users in the following way:

- If the User (Role, Team Member) has no access via standard permissions, DAC can be configured to provide or raise access levels using DAC Rules.
- MAC can conditionally deny access even if access is granted by RBAC and DAC.

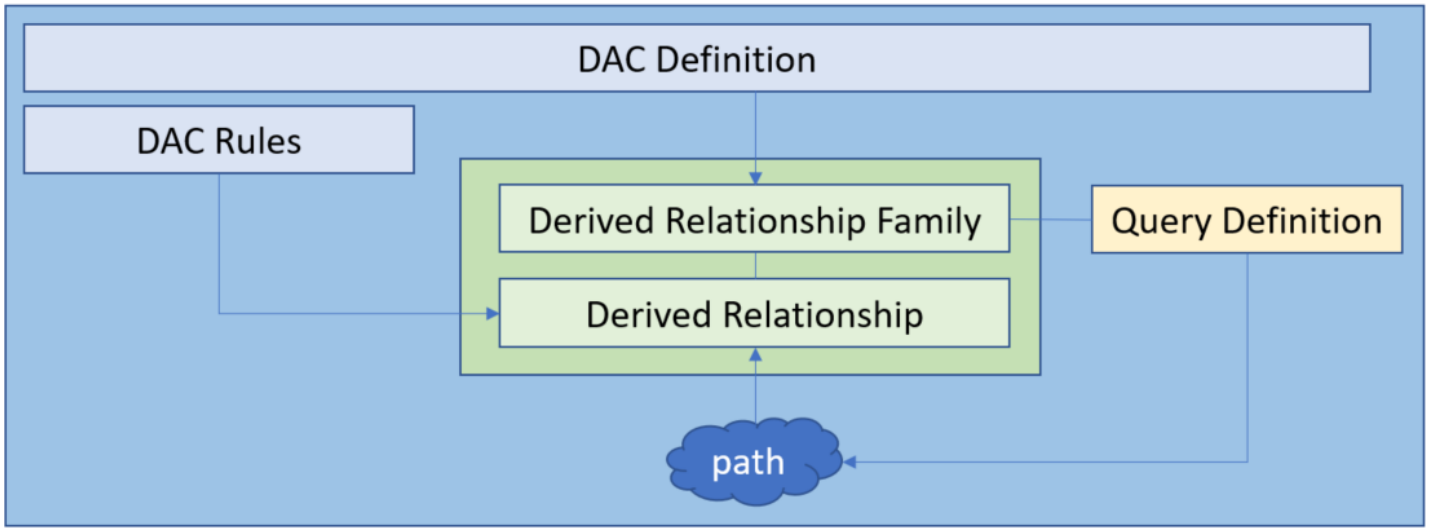


- RBAC assigns permissions directly on the item, which can only change with the lifecycle state (or authorized editing).
- DAC assigns permissions to leaf items under a subdomain if the associated rules are satisfied. The combination of RBAC and DAC Policy is “least restrictive”, meaning either one can grant permission to the current user.
- MAC imposes strict conditions on a user’s access to items by evaluating attributions on user vs. item in conditional (Boolean) expressions. If conditions are not TRUE, then access is revoked even if granted by RBAC/DAC Policies.

DAC Data Model Considerations

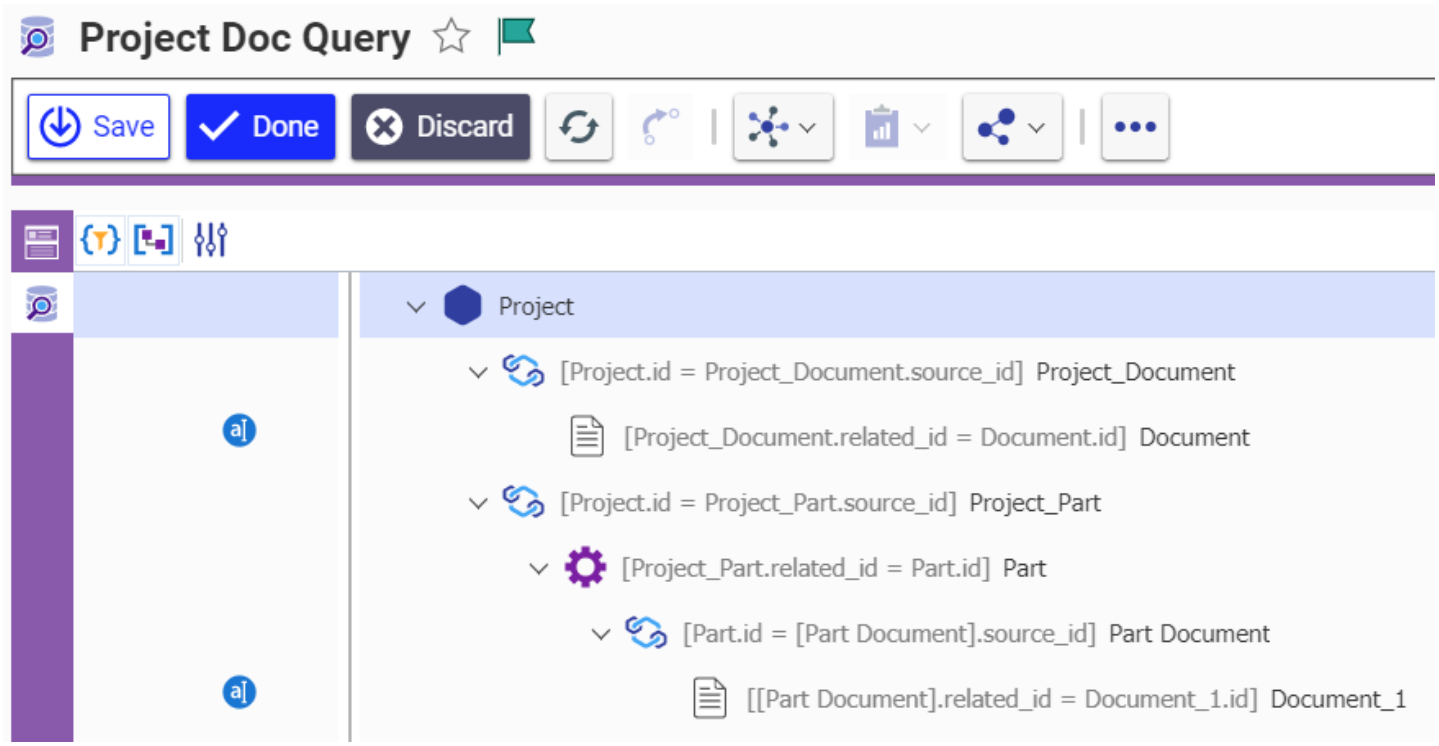
When configuring a DAC policy, it is useful to understand the DAC components particularly because it is configured “bottom-up” by administrators as you will see in Creating and Activating a DAC Domain Policy .

DAC is far from monolithic. The primary ItemType in DAC is the **DAC Definition** however it depends heavily on the **Derived Relationship** ItemType and **Derived Relationship Family** which in turn depends on the **Query Definition.** All of these must be configured before the DAC Definition can be completed.

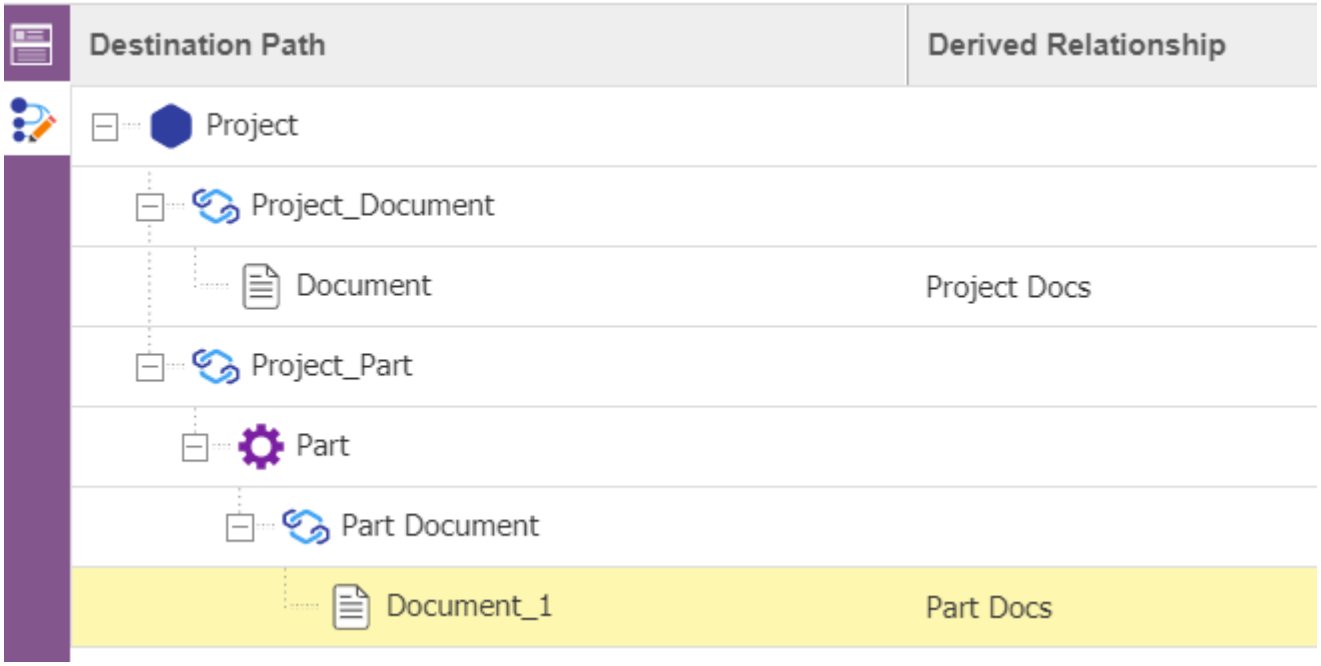


The Query Definition selects paths and endpoints from the physical relationship structure for use as Derived Relationships (DR) in a Derived Relationship Family (DRF). The DRF tracks changes to controlled items on behalf of DAC.

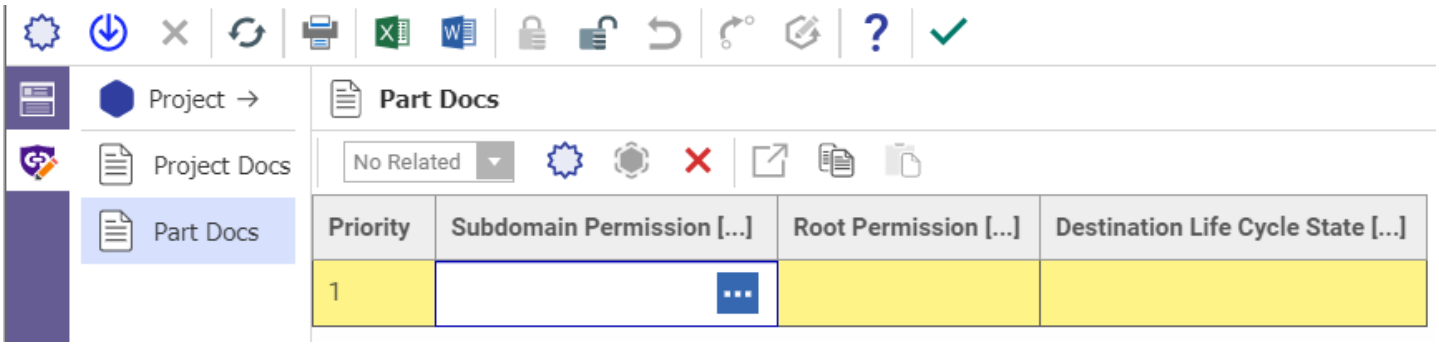
When defining a DAC Policy first create the Query Definition, use it to complete the DRF and its DRs, and then define rules for the DAC Definition.



When the Query Definition is linked to the Derived Relationship Family item, Leaf items can be selected (Root is implied) to define Derived Relationships.



Finally, the DAC Definition links to the DRF and you can assign the Subdomain Rules to each DR.



Creating and Activating a DAC Domain Policy walks through each step.

Tracking Changes to Access Controlled Items

When a DRF is first activated, an ItemType table called **unidr_Relationship** is populated with query results for every instance of Root->Leaf defined by the Derived Relationships in the family. This may take some time for large databases.

Records in this table contain references to all instances of Root and Leaf items in DAC Subdomains. After first-time initialization, the table is maintained automatically when events (add, delete, modify, etc.) occur against member items in DAC Subdomains. In this manner DAC maintains the integrity of Access Control for all item instances in the policy.

The unidr_Relationship table includes these columns:

Derived Relationship (below...)	Departure (root)	Destination (leaf)
(ID)	(ID)	(ID)
etc	etc	etc

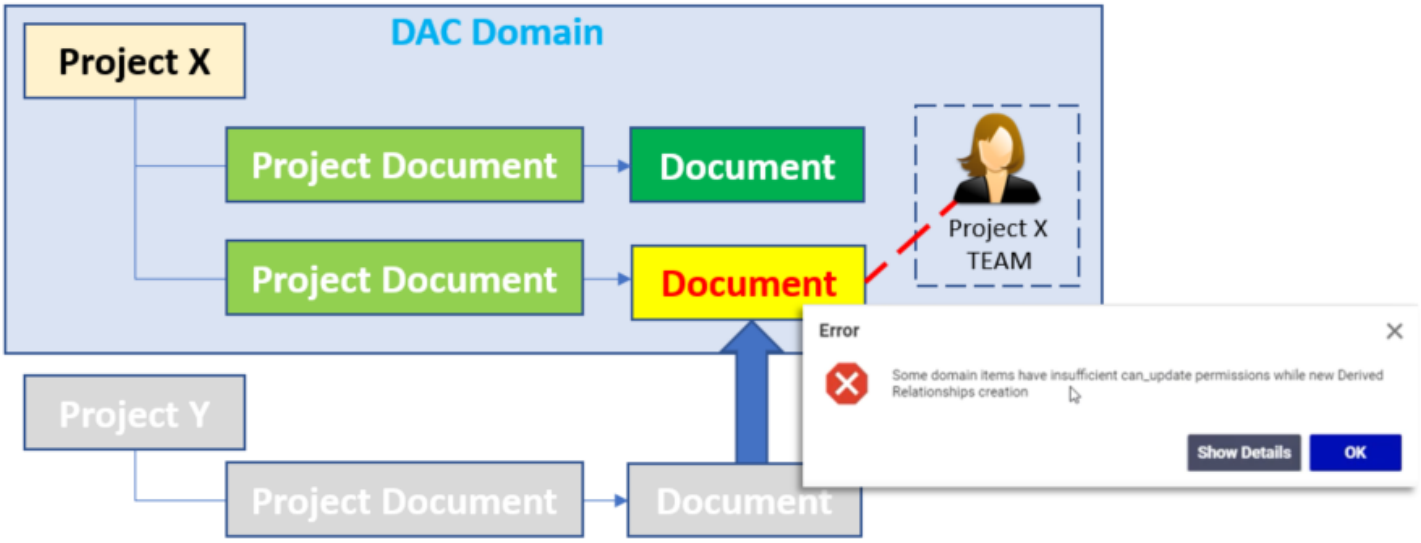
- Derived Relationship […]
  Pointer to DR under DRF linked to DAC Definition containing DAC Rules.
- Departure […]
  ID of the Root item instance of the Subdomain (e.g. Project X).
- Destination […]
  ID of Leaf item instance (e.g. Document D-123).

As of 12.0 SP5, DRF event handlers have been made configurable / customizable for those who need to execute business logic when changes to item memberships in DAC Subdomains occur.

Customizing DRF Event Handlers

Earlier we discussed the scenario where Project Team “X” gains access to an item added to the Project from another Project under DAC Policy. In some cases, business requirements prohibit the increase of access rights due to inclusion into a Domain.

For instance, a Part belonging to Project Y might be re-used by adding it to Project X. The Project X Team may suddenly gain Update rights to the Part shared with Project Y. The intent may have been to allow reuse but not open the Part from Project Y for modification by the other Project Team X.



With 12.0 SP5, DAC Domains provide a hidden ‘Events Handlers’ tab under DRF so that Administrators can either specify which Access Levels change(s) are prohibited, enable or disable such checks, or add custom logic as needed. The Administrator now has the choice to:

- Configure the DAC Definition for different Access Types (Update, Delete, etc).
- Use custom event handler method in place of the default or ,
- Add a new Event method to execute in sequence with the default Event Handler from Aras.

Configuring/customizing DRF Event Handlers is an advanced feature. Details on DRF Event Handlers is provided in the **Appendix**.

Terminology

The following terms are used in this guide to describe DAC: General Terms:

- **Access Controlled Items** - Items under DAC policy control, specifically the related items under a relationship structure that is governed by a DAC policy.
- **DAC Domain** - A logical grouping consisting of all related items sharing the same Root item whose access control policies are defined by a DAC policy.
- **DAC Subdomain** - A single path inclusive of root and leaf items in a DAC Domain.
- **Subdomain AC Policy** - the set of Access Rules that govern permissions for a Leaf item within a subdomain (path).
- **Subdomain AC Rule** - Determines which permission to grant to a Controlled Item using the configured criteria.
- **Root Item** - Top-most node of a Relationship Structure, Context Item of the Query Definition used to define Subdomains.
- **Leaf Item** - 'End' item in any path of a Relationship structure. The item whose access is controlled by DAC Policy (Access Controlled item)
- **Path** - A segment from a Relationship Structure connecting a Root Item to a Leaf Item

DAC ItemTypes:

- **DAC Definition** - An ItemType that defines a DAC policy against a domain using a Query Definition, Derived Relationship Family and DAC Subdomains. A Container for Subdomain Rules.
- **Query Definition** - A standard Aras Query Definition, used to specify paths from the controlled Relationship structure against which DAC will be applied.
- **Derived Relationship (DR)** - A Named ItemType that identifies start and endpoints as a path within a relationship structure. The start, or Departure ItemType is the root item and the Destination ItemType is the leaf item. In the DAC context it is analogous to a subdomain.
- **Derived Relationship Family (DRF)** - An ItemType that contains a Query Definition and the associated group of Derived Relationship items defined by that Query. Derived Relationships are analogous to DAC subdomains (start/end points of a path within the overall structure).
- **unidr_Relationship** - Persisted table used by DAC to track subdomains and membership of leaf items therein.