



STUDENT
TRAINING GUIDE

Defining User Access with MAC Policies

ACE23

REIMAGINE YOUR POSSIBILITIES

Agenda

Part One:

- Brief Review - Innovator Access Control

Part Two:

- Mandatory Access Control (MAC)

Part Three:

- New MAC Functionality - Multivalued Derived Attributes

Also:

- Resources / More Information

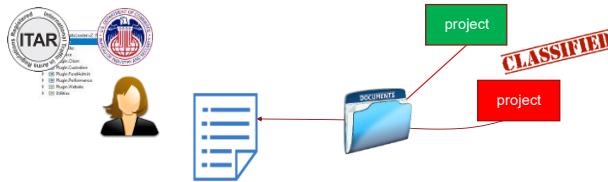
Welcome to the MAC Policies training session –

- We'll begin with a review of basic Innovator Access Control concepts, followed by the role MAC plays in the overall scheme of Access Control.
- After the introduction, we will define a MAC Policy and Activate it to test it.
- We will then proceed to review and implement the latest and most powerful capability of MAC known as Multi-valued Derived Attributes.



Perform hands-on exercises on provided machines

Part One: Innovator Access Control Review



Access Rights

By now you're familiar with Itemtype Permissions, which are based on *Access Rights* granted to Identities: **Get, Update, Delete, Discover, Show Warnings, Change Access**

- They are required on every Itemtype
- Changes in Lifecycle State often used to change Permissions automatically (for instance when Releasing a Part or Document)
- Essentially, Permissions assign Access Rights to Identities and Roles

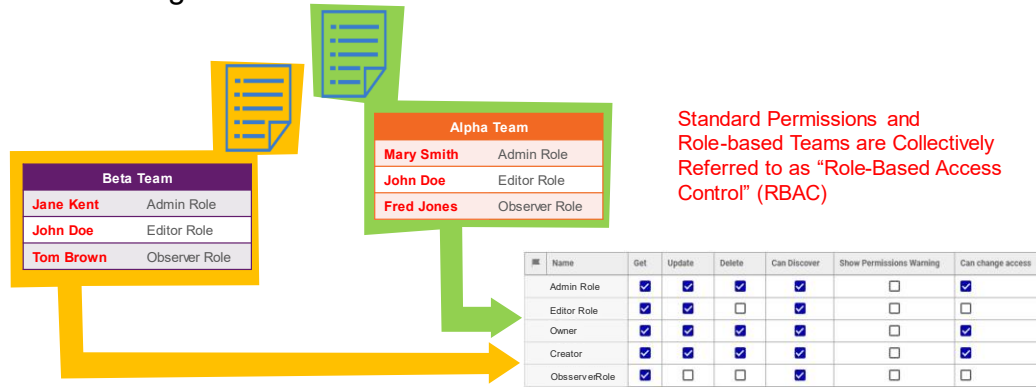
| Name | Get | Upd... | Dele... | Can Discover | Show Permis... | Can change a... |
|---------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-------------------------------------|
| Aras PLM | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Manager | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| Owner | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| Creator | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| All Employees | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

The key point here is that Access Rights are Integral to all Access Control schemes:

- ✓ Permissions
- ✓ Teams
- ✓ Domain Access Control (DAC)
- ✓ **Mandatory Access Control (MAC)**

Role-Based Teams

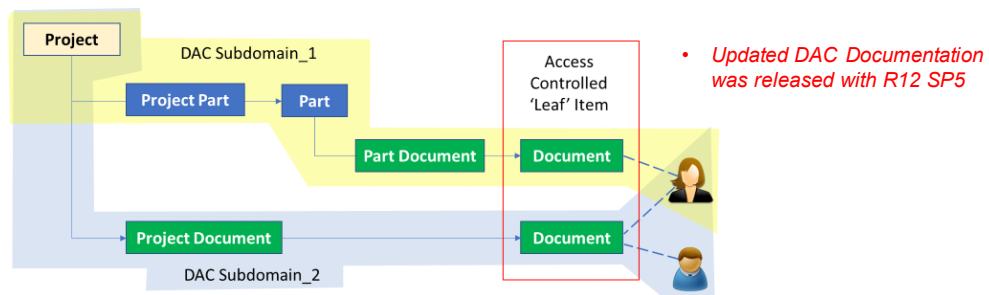
- Teams enable Run-time Assignment of Identities to Roles
- A Team Item on a Document Item resolves abstract roles into specific Identities for Access Rights



Teams allow for a 'late binding' approach to the Identity (left column) in an Access Right's row. The Identity is abstracted as a Role, replacing the Identity in a Permission. The Role is resolved to a specific Identity in a Team Item instance attached to the Item.

Domain Access Control (DAC)

- DAC determines which permission to grant based on two things:
 - Item's **Relationship** to a root item – for instance a Project, Territory, Department or other " **Domain**"
 - Matching criteria between Item and Root Item including Lifecycle State and Parent Permission
- Access Rights are conditionally **granted** by DAC or RBAC



Domain Access Control grants access using rules applied to an item occurring within a relationship structure or "Domain". Moving items in or out of a DAC domain may change its access. Access is granted in an additive manner to RBAC permissions – it can elevate but not restrict access. DAC also supports Teams and supports Lifecycle-driven permissions.

Mandatory Access Control (MAC)

- MAC on the other hand evaluates Properties on the Current User Item and/or the Item whose access is being requested in order to control Access Rights
- Boolean Expressions define the Conditions of Access
- MAC **revokes** Access – it does not grant it

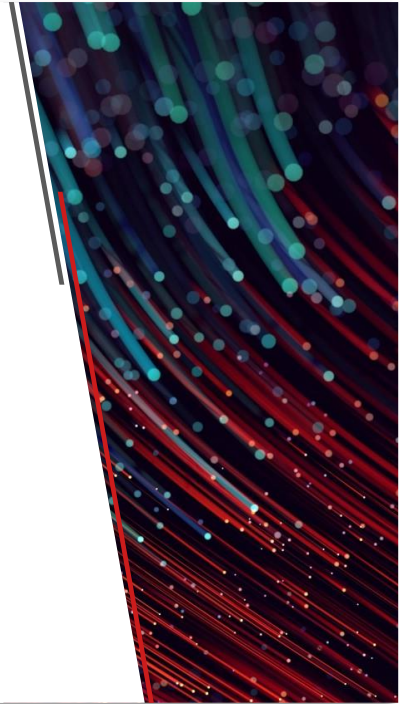


While RBAC (Permissions and Teams) and DAC can grant access, MAC can only revoke access. Therefore, if there was no access to an Item to begin with then passing the MAC test will not matter – it will remain inaccessible. Conversely, if access granted to an item was granted by any preceding means, failing the MAC test will revoke access.

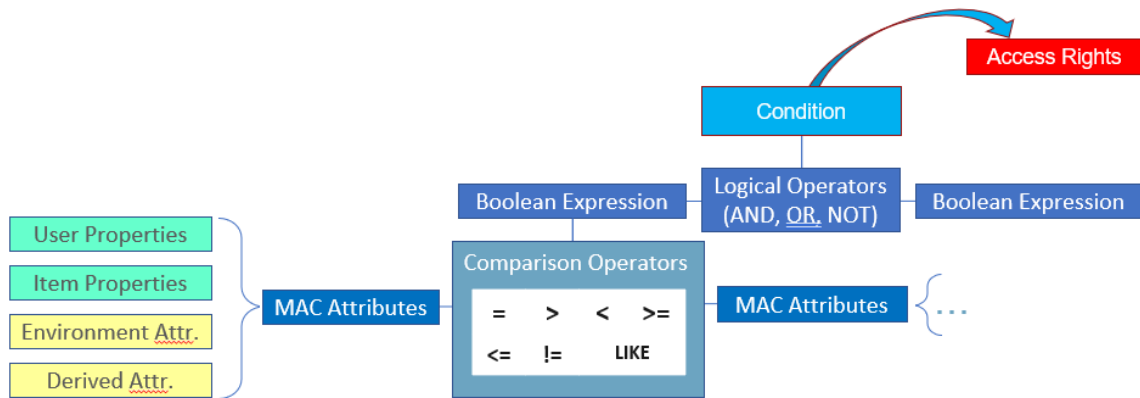
Summary

- Access Control is enforced via Access Rights per Identity (or Role)
 - Permission changes are often triggered by LifeCycle State Transitions
- RBAC and DAC can **grant** access to Items
- MAC can **deny** access even if previously granted by RBAC/DAC

Part Two: How MAC Works



The Anatomy of a MAC Policy:



MAC Attributes

- The Values derived from Items, Users, Environment Variables, and a special construct 'Derived Attributes' (covered later).
- Used as Operands in MAC Boolean Expressions

Boolean Expressions

- TRUE/FALSE expressions involving Attributes compared against each other
- Comparison Operators provided (see above)

Condition

- Group of one or more Boolean Expressions combined using logical AND, OR, NOT into one 'Condition' that can be applied to an Access Right

Access Rights

- Get, Update, Discover, Delete, Show Permissions Warning; Common to all Innovator Access Control schemes (RBAC, DAC, MAC)

Rule

- Condition + Access Right = Rule
 - A collection of expressions defines a Condition; when applied to Access Rights it defines a Rule. A Policy is the set of rules against all affected Itemtypes.

Attributes

MAC overrides any existing item access (RBAC/DAC) by applying Conditions against Access Rights. Conditions are boolean expressions that can be very simple, or as complex as you may need. These boolean expressions evaluate **Attributes** which are the values used in MAC expressions. Attributes can be derived in many useful ways – which is a key feature of MAC. Boolean expressions can reference CurrentItem, CurrentUser Attributes which may be:

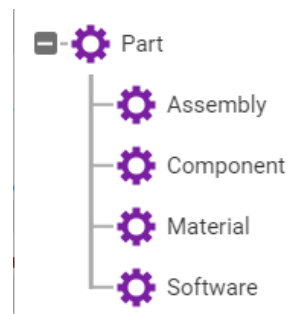
- Property-based attribute obtained from the Item being accessed – CurrentItem.<Property_name>
- Property-based attribute from the User making access request – CurrentUser.<Property_name>
- Environment Attribute – dynamically generated on invocation via Method execution
- Derived Multivalued Attribute – created using the Derived Attribute Definition item
- xClass or xProperty reference used as an attribute
- Attributes can also be constants (hard-coded values).

Simple Attributes in Conditional Expressions

Perhaps the simplest form is a condition on the current Item (type Part) based on one of its existing Attributes, Class Structure:

```
CurrentItem.Classification="Software"
```

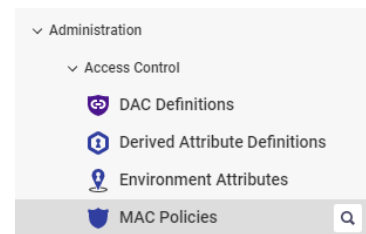
Such an expression could be applied to an Access Right, for instance "Get".







Try It:

Use an existing MAC Policy.

- 1) Navigate to Administration/Access Control/MAC Policies
- 2) Open MAC Policy 'MP001'
- 3) Use the Sidebar menu to open the Rules Editor window, examine the rule
- 4) Navigate to Design/Parts and [Search] for all Parts
- 5) Activate the MAC Policy from the [...] More toolbar menu
- 6) Search for all Parts again
- 7) Add 'Administrator' to the Exempt Identities tab on the MAC Policy
 - a. First Deactivate the MAC Policy



- b. Generate a New Version, then Edit
 - c. On the 'Exempt Identities' tab, add Administrators group
 - d. Save the MAC Policy
- 8) Search all Parts again
- 9) Deactivate '**MP001**'

| | |
|--|-------------|
|  Navigate > | Delete > |
|  Reports > | Activate |
|  Share > | Deactivate |
|  More > | New Version |

Item Properties vs. User Properties as MAC Attributes

A more practical use case is that in which the attributes of the current User are compared against the attributes of the item the User is trying to access.

The User Itemtype and the Part Itemtype have both been assigned a new List Property 'Security Level' that has the following levels as Label/Values:

| Label ↑ | Value |
|-------------------------|-------|
| Controlled Unclassified | 1 |
| Public Trust Position | 2 |
| Confidential | 3 |
| Secret | 4 |
| Top Secret | 5 |
| Compartmented | 6 |



Try It:

| Name | Condition |
|----------------------|---|
| Security Level Match | CurrentItem.[Security Level] = CurrentUser.[Security Level] |

- 1) Navigate to Administration/Access Control/MAC Policies
- 2) Open MAC Policy 'MP002'
- 3) Use the Sidebar menu to open the Rules Editor window
- 4) Navigate to Design/Parts and [Search] for all Parts
- 5) Activate 'MP002' from the [...] More toolbar menu
- 6) Search for all Parts again
- 7) Activate 'MP001' as well
- 8) Search all Parts again
- 9) Deactivate both from the main search grid using the RMB 'More' menu

Access if User Level Higher than Item Level Only

In the following use case, the Item can only be accessed (Get, Discover) if the Security Level of the Item is less than or equal to that of the User. We simply need to change the operator from '=' to '<='.

Try It:

| Name | Condition |
|----------------|--|
| only less than | CurrentItem.[Security Level] <= CurrentUser.[Security Level] |

- 1) Navigate to Administration/Access Control/MAC Policies
- 2) Open MAC Policy 'MP002'
- 3) Deactivate if Active / Create a new Version
- 4) Edit the MAC Policy and use the Sidebar menu to open the Rules Editor window
- 5) Change the operator from '=' to '<=' as shown
- 6) Save and Activate
- 7) Edit Part Security Levels to various combinations with User Security Levels to test

You can use the following search criteria to isolate results. Modify the Admin User's security level, modify some Part security levels in various combinations to test.

 **Parts**  

 |

| ItemType | Property | Operation | Criteria[...] |
|----------|----------------|-----------|---------------|
| Part | Security Level | not null | |

Helper Method “IsMemberOf()” Example

An option to Attributes derived from Properties would be to check User membership in special group Identities. This would eliminate the need to implement Property schemes on User Items, which requires Root login, and can become complex over time. Such a Boolean Expression might look something like this:

“IsMemberOf()” is a Helper Method provided by MAC, another one of the many ways MAC can derive Attributes for use in expressions.

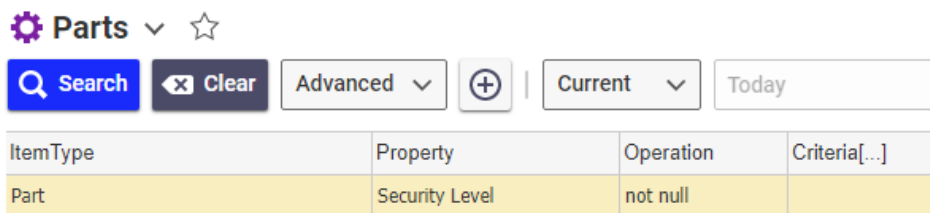
This MAC Expression would solve to TRUE for any Item that has no Security Level set. But if it did have Security Level set, then the User must be a member of ‘Authorized Group’ to solve to TRUE.

The Expression would be applied to Access Rights to enforce the Rule against the specified Itemtype, in this case Parts.

*(CurrentItem.[Security Level] IS NULL)
OR
((CurrentItem.[Security Level] >= 1) AND CurrentUser.IsMemberOf('Authorized Group'))*

Try It:

- 1) Open MAC Policy MP003
- 2) Activate it
- 3) List Parts with ‘Confidential’ Security Level



The screenshot shows a search interface for 'Parts'. At the top, there is a search bar with a 'Search' button, a 'Clear' button, and a dropdown menu set to 'Advanced'. Below the search bar is a table with the following columns: 'ItemType', 'Property', 'Operation', and 'Criteria[...]'. The table contains one row: 'Part', 'Security Level', 'not null'.

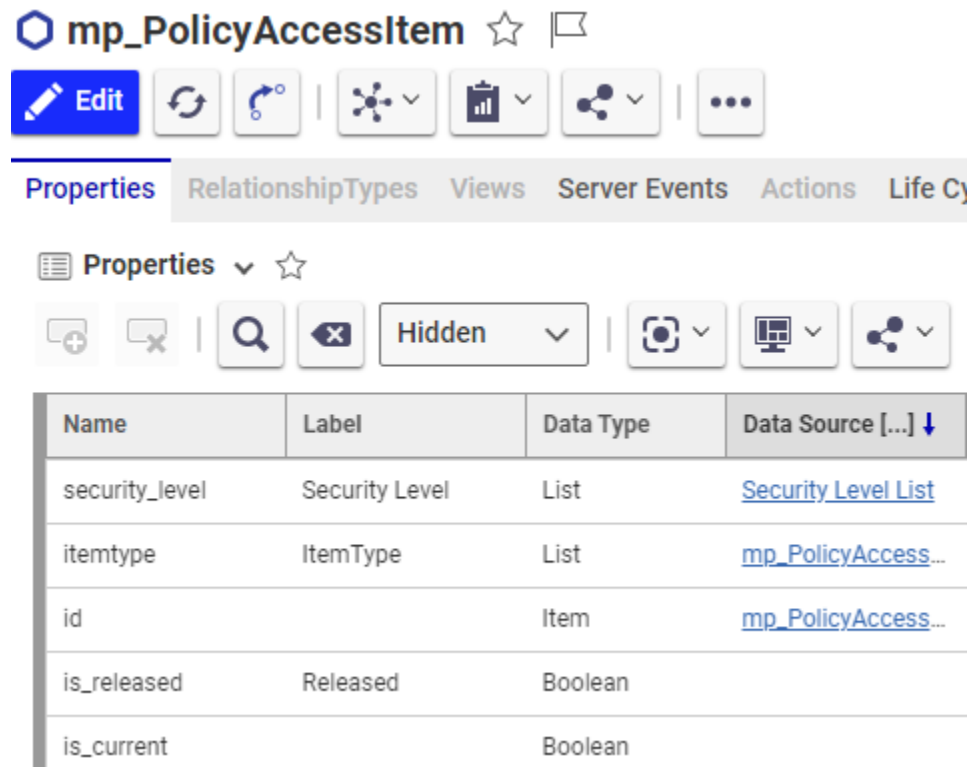
| ItemType | Property | Operation | Criteria[...] |
|----------|----------------|-----------|---------------|
| Part | Security Level | not null | |

- 4) Add admin to Identity Group ‘Authorized Group’
- 5) Log out, Log back in to reset group membership cache
- 6) List Parts again

Enabling Itemtype Properties for Use as MAC Attributes

Itemtype Properties to be used as MAC Attributes must be added to the Properties tab of a special MAC Itemtype called **mp_PolicyAccessItem**. This was done for 'Security Level' in preparation for these exercises.

For example:



The screenshot shows the configuration interface for the **mp_PolicyAccessItem** class. The 'Properties' tab is active, displaying a table of properties. The table has four columns: Name, Label, Data Type, and Data Source [...].

| Name | Label | Data Type | Data Source [...] |
|----------------|----------------|-----------|-------------------------------------|
| security_level | Security Level | List | Security Level List |
| itemtype | ItemType | List | mp_PolicyAccess... |
| id | | Item | mp_PolicyAccess... |
| is_released | Released | Boolean | |
| is_current | | Boolean | |

Once added, a Property can be referenced as an Attribute in any MAC Policy Expression. Properties added to **mp_PolicyAccessItem** must exactly match the properties of the ItemTypes that the MAC Policies are being applied to. All ItemTypes that the MAC Policy is being applied to must have the property being referenced, otherwise validation will fail when the Admin attempts to save the Policy. The following data types are supported:

| | | | | | | | |
|--------|---------|-------|---------|---------|------|------|------|
| String | Integer | Float | Decimal | Boolean | Date | Item | List |
|--------|---------|-------|---------|---------|------|------|------|

Try It:

- 1) Note that the 'Security Level' Property was added in preparation for this class.
- 2) Edit the Document Itemtype to add a Property 'Security Level'
 - a. Type List
 - b. Source 'Security Level List'
- 3) Update the Document Form to allow selection of a Security Level
- 4) Click Done

Implementing New MAC Policies

Aras Innovator MAC Policies are used to control user access to Items through a set of MAC Policy Rules. MAC Policy Rules control access rights for Get, Update, Delete, Can Discover, and Show Permission Warning to the set of ItemTypes which the MAC Policy is applied to.

Creating a New MAC Policy

You can find MAC Policies in Aras Innovator by clicking Administration → Access Control → MAC Policies in the TOC. The following menu appears:

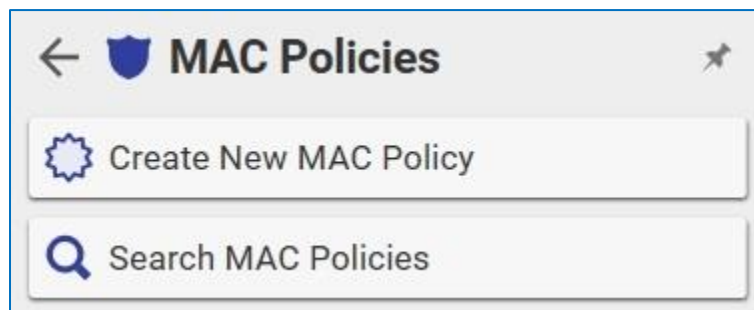

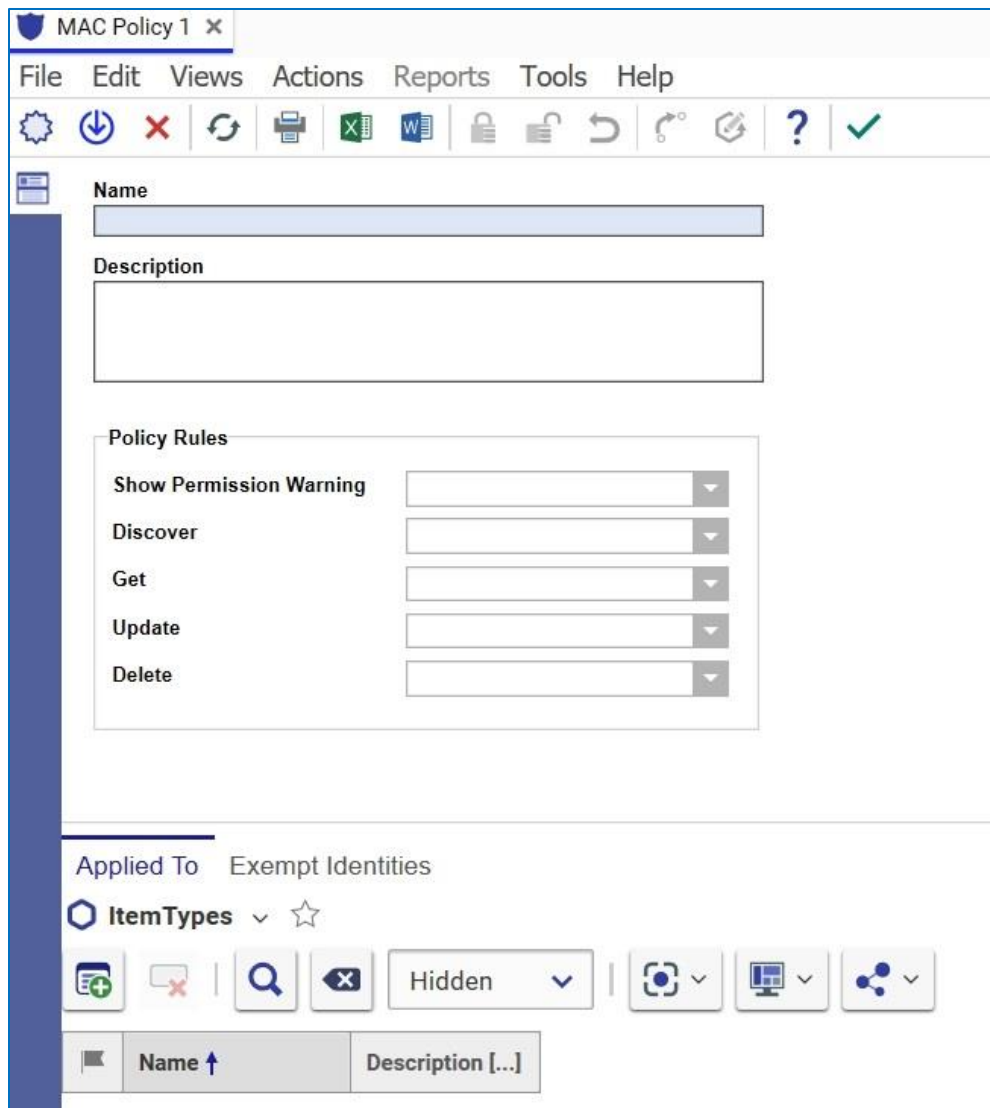


Figure 1.

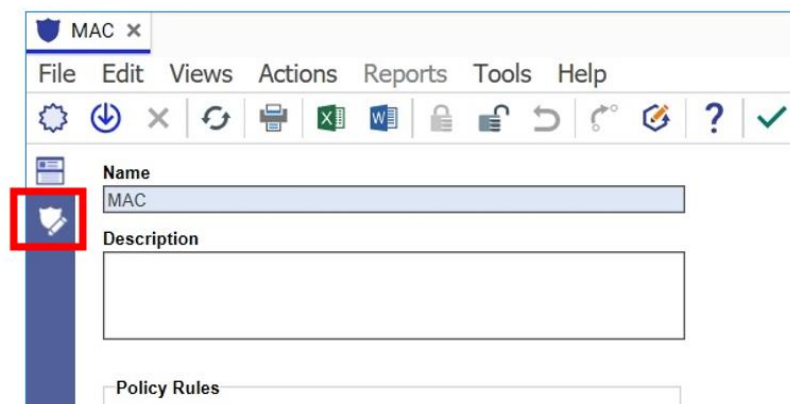
Only users with Administrative permissions have the ability to create MAC Policies. Once you create a MAC Policy, you must specify the Name and save it. Once you do that, you can create MAC Policy rules. Use the following procedure:

Try It:

- 1) Navigate to Administration/Access Control/MAC Policies
- 2) Click Create New MAC Policy. The following screen appears (next page )



- 3) Enter the Policy Name '**MP004**' in the **Name** field and add the **Document** Itemtype in the 'Applied To' tab
- 4) click Save.
- 5) The MAC Expression Editor appears in the Sidebar menu (below):



Expression Editor

- MAC provides an Expression Editor
- Supports Logical AND, OR, NOT and Comparison Operators
- Syntax Checking

| Operation | Name | Usage | Meaning |
|-----------|--------------------------|--------------------------|--|
| = | Equals | valueRef1 = valueRef2 | TRUE if left value is equal to right value. |
| > | Greater Than | valueRef1 > valueRef2 | TRUE if left value is greater than right value. |
| < | Less Than | valueRef1 < valueRef2 | TRUE if left value is less than right value. |
| >= | Greater Than or Equal To | valueRef1 >= valueRef2 | TRUE if left value is greater than or equal to right value. |
| <= | Less Than or Equal To | valueRef1 <= valueRef2 | TRUE if left value is less than or equal to right value. |
| != | Not Equal To | valueRef1 != valueRef2 | TRUE if left value is not equal to right value. |
| LIKE | Like | valueRef1 LIKE valueRef2 | True if left value matches the right value (pattern). The syntax for the "LIKE" operator is exactly the same as in Transact-SQL. |

| Operation | Name | Usage | Meaning |
|-----------|------|------------------------|---|
| && | AND | valueRef1 && valueRef2 | TRUE if both valueRef1 and valueRef2 are TRUE. |
| | OR | valueRef1 valueRef2 | TRUE if either valueRef1 or valueRef2 are TRUE. |
| ! | NOT | !valueRef1 | TRUE if valueRef1 is FALSE. |

Logical Operators (AND, OR, NOT)



We have used the editor to modify condition expressions in previous exercises. We will now create new ones. The following syntax rules apply to Condition expressions:

- Values are case sensitive.
- String literals text must be enclosed between quotation marks ('text'), quotation escapes with back slash ('\').
- A Constant can act as an operand when you use it in a comparison. Otherwise a Constant is a string type.
- Operators and precedence are the same as those used in SQL languages.
- Operators are not case sensitive.
- Arithmetic operators are not supported.
- Parentheses can be used to override operator precedence.
- Comparing two strings follows Transact-SQL rules.

Supported Comparison Operators for MAC Expressions

| Operation | Name | Usage | Meaning |
|-----------|--------------------------|------------------------|---|
| = | Equals | valueRef1 = valueRef2 | TRUE if left value is equal to right value. |
| > | Greater Than | valueRef1 > valueRef2 | TRUE if left value is greater than right value. |
| < | Less Than | valueRef1 < valueRef2 | TRUE if left value is less than right value. |
| >= | Greater Than or Equal To | valueRef1 >= valueRef2 | TRUE if left value is greater than or equal to right value. |

| Operation | Name | Usage | Meaning |
|-----------|-----------------------|--------------------------|--|
| <= | Less Than or Equal To | valueRef1 <= valueRef2 | TRUE if left value is less than or equal to right value. |
| != | Not Equal To | valueRef1 != valueRef2 | TRUE if left value is not equal to right value. |
| LIKE | Like | valueRef1 LIKE valueRef2 | True is left value matches the right value (pattern). The syntax for the “LIKE” operator is exactly the same as in Transact-SQL. |

Note: If the Policy Rule uses a Property on the User ItemType and a user is able to edit their own User Item, then the user is capable of changing their level of access to any of the ItemTypes that a MAC Policy is applied to. This can also apply if the Edit control is not restricted in a MAC Policy and other Permissions are based on editable Properties.

! It is best to avoid using Properties on the User ItemType where it is possible and suitable to use group membership or derived attributes instead (covered later).

This policy is responsible for restricting access to User, Identity, and Alias items based on defined User Visibility Rules items and should NOT be modified. The User Visibility Policy MAC Policy and User Identities Derived Relationship Family must both be in the Active state in order for User Visibility Rules to be enforced.

Try It (continued)

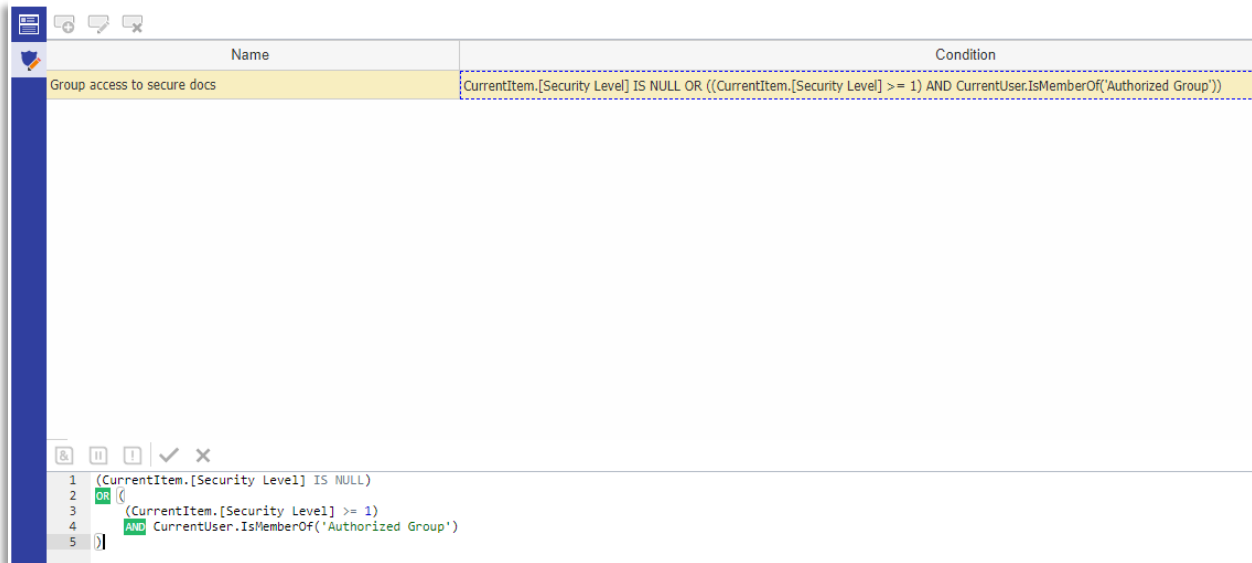
5) Add the following expression:

```
(CurrentItem.[Security Level] IS NULL)
OR
((CurrentItem.[Security Level] >= 1) AND CurrentUser.IsMemberOf('Authorized Group'))
```

6) Name the expression – you’ll use this name to assign to Access Rights on main form (E.g., “Secure Doc Access”)

7) Use the green checkmark to save it, which puts the condition into the top bar of the editor

| Condition |
|--|
| CurrentItem.[Security Level] IS NULL OR ((CurrentItem.[Security Level] >= 1) AND CurrentUser.IsMemberOf('Authorized Group')) |



Note that Condition editing is done in the lower pane, and the upper frame displays saved Conditions previously created within this MAC Policy Item. There can be multiple condition expressions in a policy.

Available Helper Methods

The following table lists available methods that you can use within the Condition statement of a Policy Rule:

| Method | Response |
|---|---|
| CurrentUser.IsMemberOf(<Identity Name>) | Returns true if the current user is a member of a (non-system) Identity <Identity Name>, otherwise it returns false. |
| CurrentUser.IsMemberOf(Property<Item>) | Returns true if the current user is a member of the Item Property of type Identity. For example: <i>CurrentUser.IsMemberOf(CurrentItem.identity_id)</i> |
| String.Contains(<StringToSearch>, <SearchForString>) | Returns true if <SearchForString> is a substring of <StringToSearch>, otherwise it returns false. |
| CurrentItem.HasUserVisibilityPolicyAccess() | Returns true if the current user has access to the current item based on the active User Visibility Rules. This function can only be applied to User, Alias, Identity Item Types. |

Applying Policy Rules to Access Rights

Once you create Conditions, you may assign them to specific Access Rights. There are selection fields for each.

Select a Condition from the dropdown list to apply that Condition to the Access Right. The user is denied that Access Right if the selected conditions are not met (resolves to FALSE).

The screenshot shows the configuration page for a MAC Policy named 'MP004'. The description is 'MAC control for secure documents'. Under the 'Policy Rules' section, there are five dropdown menus for different access rights: 'Show Permission Warning', 'Discover', 'Get', 'Update', and 'Delete'. The 'Discover' dropdown is currently selected and shows 'Group access to secure docs' as the chosen rule.

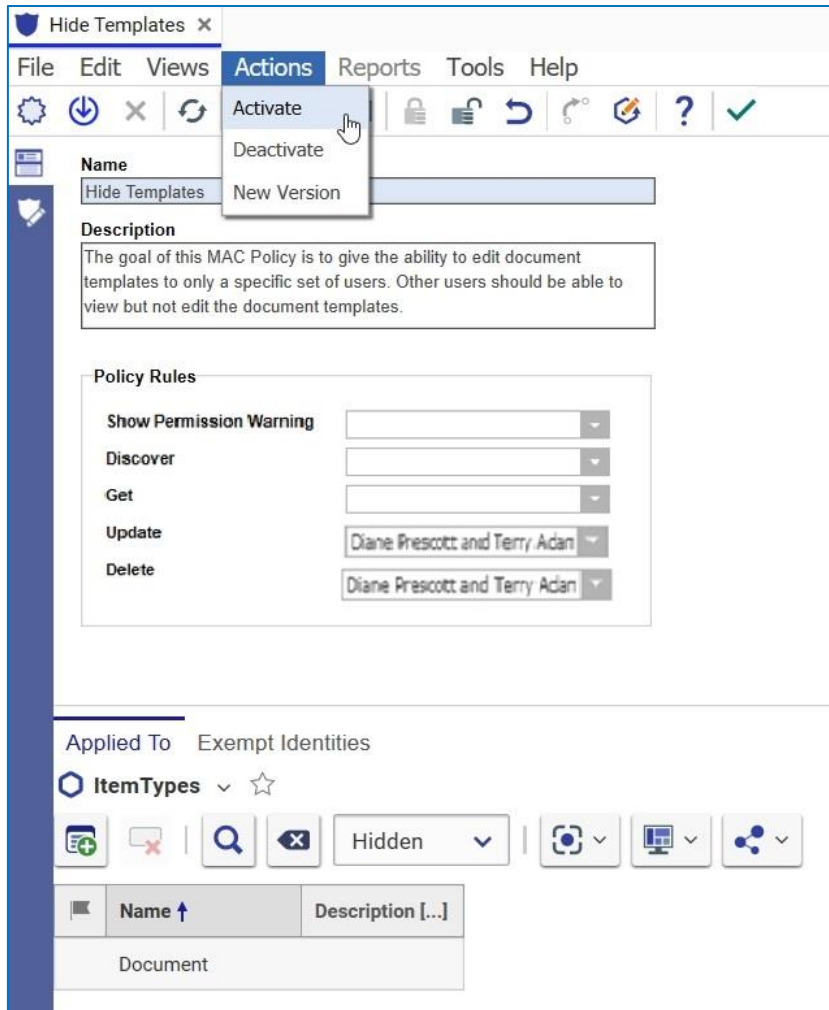
Note: If a Rule is not selected for an access right, then the MAC Policy will not apply any restrictions to that access right.

Standard Aras Innovator role-based permissions and all applicable MAC Policy rules must grant access to an Item before a user is able to access it.

Applying MAC Policies

Once you establish the Policy Rules, you must assign the MAC Policy to the desired ItemTypes and then activate the MAC policy. Assigning the MAC policy to ItemTypes is done by adding them to the Applied To Relationship.

To activate a MAC Policy, unclaim the policy and select the 'Activate' Action. When Activating a MAC Policy, all users except for the admin activating the MAC Policy should be logged off and prevented from accessing the system until the MAC Policy is activated.



Try It:

- 1) Add the condition to Discover and Get Access Rights at a minimum
- 2) Save the MAC Policy Activate it
- 3) Test on **Documents**
 - a. Set Security Level on some

Question: Why isn't it necessary to add the condition to Update or Delete?

Updating MAC Policies

Once activated, a MAC Policy can be deactivated or versioned. The 'Deactivate' action sets a MAC Policy to the 'Inactive' state. When a MAC Policy is 'Inactive,' the Policy Rules are not applied to control access.

In order to modify a previously active MAC Policy, the Policy must be versioned through the 'New Version' Action. The previous revision of the MAC Policy is promoted to the 'Archived' state only after the new version of the Policy is activated.

Exempt Identities

Each MAC Policy includes a list of "Exempt identities." If the user making an access request has an identity that is included in the "Exempt identities" list, MAC Policy rules are not applicable to the request. (We used this in an earlier exercise)

Using xClasses and xProperties in MAC Policy Conditions

You can use xProperties that are associated with the mp_PolicyAccessItem ItemType as item attributes in MAC Policy conditions. You should specify supported xProperties in the Allowed xProperties relationship tab in the mp_PolicyAccessItem.

You can also use xProperties that are associated with the User ItemType as user attributes. Using xProperties in MAC Policy conditions may cause some performance penalties but it has the advantage of being able to specify access control for an item xProperty independently from access control for the item itself (and therefore all the item regular properties).

Warning DO NOT edit xProperty definitions that are being used in active MAC policies. Doing so results in a system failure.

In MAC Policy conditions you can also check if an item or a user is classified by an xClass using the built-in functions described below.

Using Methods to Verify Item Classification

The following table lists methods that enable you to check in MAC policy conditions to see if an item or user is classified using a particular xClass or xProperty:

| Method | Response |
|--|---|
| CurrentItem.IsXPropertyDefined(<xPropertyName>) | Returns true only when the <xPropertyName> is defined on CurrentItem. |
| CurrentItem.IsClassifiedByXClass(<xClassName>) | Returns true only when the CurrentItem is classified by <xClassName>. |
| CurrentUser.IsXPropertyDefined(<xPropertyName>) | Returns true only when <xPropertyName> is defined on Current.User. |
| CurrentUser.IsClassifiedByXClass(<xClassName>) | Returns true only when CurrentUser is classified by <xClassName>. |

For more information about xClasses and xProperties, refer to the Extended Classification guide, and the MAC Policies released documentation.

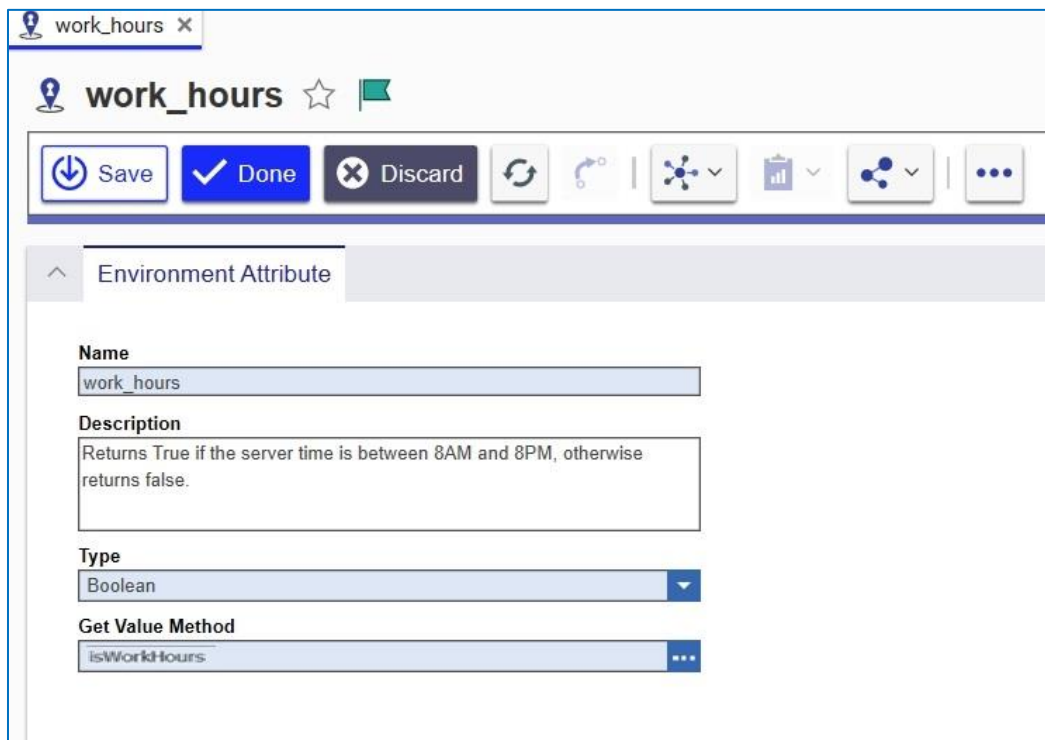
Defining Environment Attributes


Environment attributes grant a user certain access rights to an Item based on specific circumstances such as the geographical location or the time an access request is made. For example, if the user makes the request outside of work hours, the request is denied. If the same user makes the same request during work hours, their request is accepted.

You can find and create Environment Attributes in Aras Innovator by clicking **Administration** → **Access Control** → **Environment Attributes** in the TOC. The following menu appears:



1. Click **Create New Environment Attribute**. A blank Environment Attribute window appears:
2. Enter the appropriate information in the Name, Description, Type and Get Value Method fields.

A screenshot of a configuration window for an environment attribute named "work_hours". The window has a title bar with a location pin icon and the text "work_hours x". Below the title bar is a toolbar with icons for Save, Done, Discard, Refresh, Undo, and other actions. The main content area is titled "Environment Attribute" and contains four fields: "Name" with the value "work_hours", "Description" with the text "Returns True if the server time is between 8AM and 8PM, otherwise returns false.", "Type" with a dropdown menu set to "Boolean", and "Get Value Method" with a dropdown menu set to "isWorkHours".

3. Click  to save and unclaim the attribute.

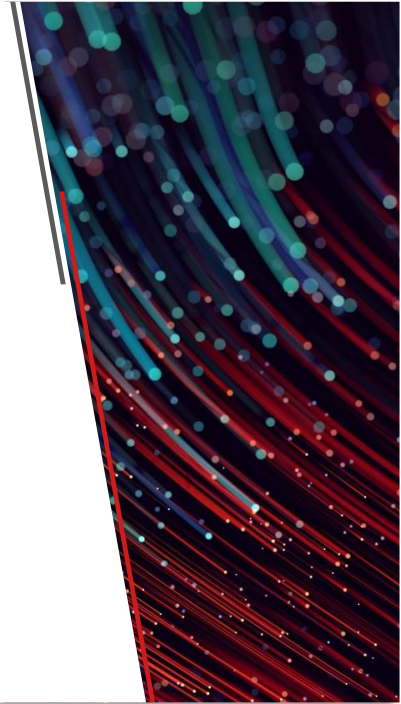
Each attribute has a unique Name, Type, and custom method (specified in the “Get Value Method” property) that is used to get the attribute value to use in an expression.

An environment security attribute with the name <attr_name> is referenced in a Condition expression as ‘\$<attr_name>’. The supported data types for Environment Attributes are Boolean, String, and Integer.

Here is an example of a method that can be specified in the Get Value Method field (cut and paste into method):

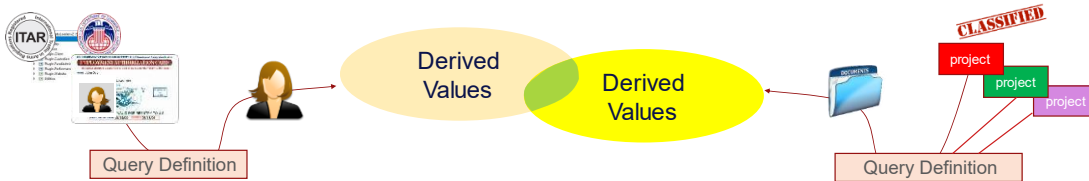
```
//MethodTemplateName=CSharp:Aras.Server.Core.AccessControl.Environment
AttributeMethod;
var startWorkTime = new TimeSpan(8, 0, 0);
var endWorkTime = new TimeSpan(20, 0, 0);
var currentDateTime = DateTime.Now;
var isWorkDay = DayOfWeek.Monday <= currentDateTime.DayOfWeek &&
currentDateTime.DayOfWeek <= DayOfWeek.Friday;
var isWorkTime = startWorkTime <= currentDateTime.TimeOfDay &&
currentDateTime.TimeOfDay <= endWorkTime;
var isWorkHours = isWorkDay && isWorkTime;
attribute.SetValue(isWorkHours);
```


Derived Multi-valued Attributes ★ New



Introducing MAC Derived Attributes

- Like regular MAC Attributes, they are used to define policy Rules.. with powerful new features:
 1. Property Values from **Related or Referencing Items** can also be used in expressions
 - e.g., "Check if the Projects Referencing this Item Require Extra Security"
 2. Rather than single Property values, multi-valued sets or **collections** are supported
 - e.g., "If [all User's qualifications] vs. [all qualifications required by Projects where used] has no matches then REVOKE access"

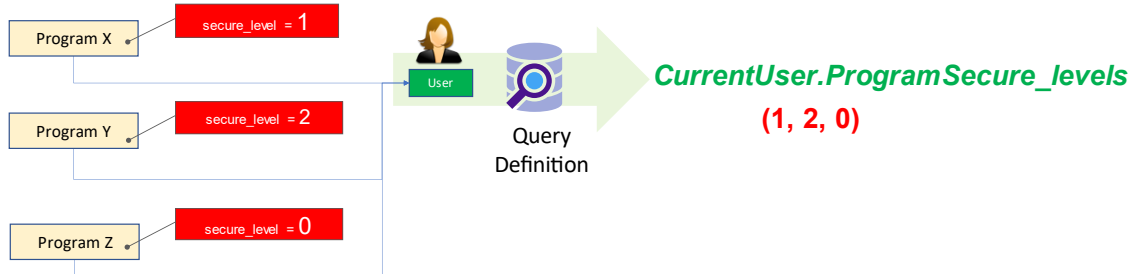


Derived Attributes Use Query Definitions

- An Integrated Query Specifies a "Path" to a Related/Referencing Item, and the Property to derive a set of values to use as multi-valued MAC Attribute

For example:

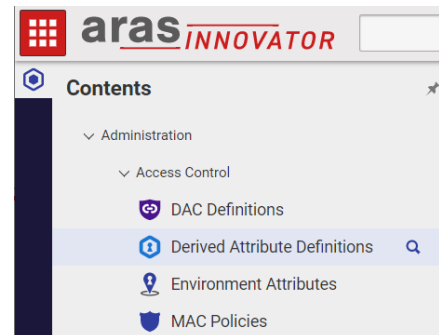
- "Query for Set of all Security Level Properties from all Programs that the Current User is Assigned to"
- *CurrentUser.ProgramSecure_levels = (1,2,0)*



Derived (Multi-Valued) Attributes

These special attributes are configured independently (like Environment Attributes) and become available in the Condition Editor when successfully configured. Derived Attribute Definitions reside under Administration/Access Control in the table of contents.

Derived Attributes use an embedded **Query Definition** to specify the Path from the Root (CurrentItem) to a related Leaf Item and a Target Property from which the Derived Attribute values are extracted.



Configuring a Derived Attribute

A Derived Attribute has the following structure:

The screenshot shows the configuration interface for a derived attribute. It is divided into two main sections: 'Derived Attribute Definition' and 'Attribute Queries'.

Derived Attribute Definition:

- 1** **Name:** User Restriction for Part Doc
- 2** **Datatype:** Integer
- 3** **Description:** Conditional Access Given Restriction Level of Parent Part -> Document

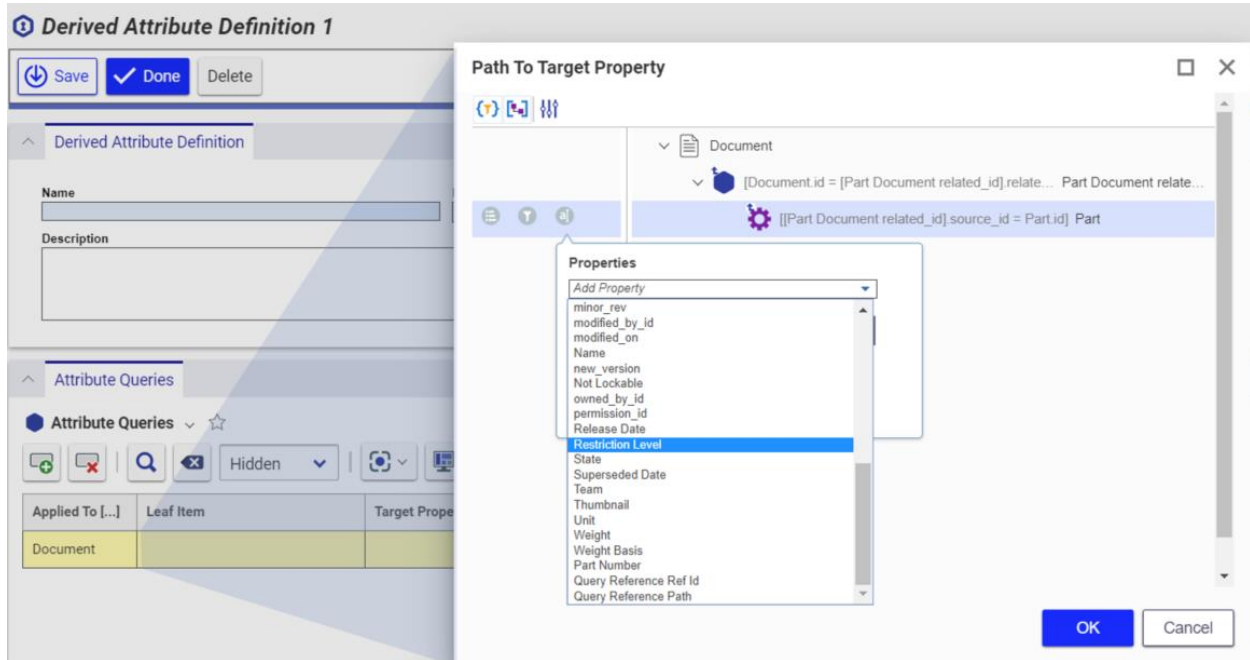
Attribute Queries:

| Applied To [...] | Leaf Item | Target Property |
|------------------|-----------|-------------------|
| Document | Part | restriction_level |

4 points to the 'restriction_level' property in the table.

| | | | | | | | | | | |
|---------------------------------|---|--|---|------------------|--|---------------------------|--|------------------|------------------------|--|
| 1 | Name – the Symbol by which this derived attribute is referenced in expressions (Required). Enclose spaces in [square brackets] when used in an expression. | | | | | | | | | |
| 2 | Datatype of values contained in a derived attribute collection (Required). Must match Target Property type, and vice-versa. | | | | | | | | | |
| 3 | Text Description of Attribute for reference. | | | | | | | | | |
| 4 | Attribute Queries - related items that define how to query the values for this 'derived' attribute | | | | | | | | | |
| <i>First-time Entry Only</i> | <table border="1"> <tbody> <tr> <td>“Applied To” Itemtype</td> <td>The Itemtype that will define the scope of “CurrentItem” in Boolean expressions; also the Root Item of the Path defined by the <i>Embedded Query Definition</i>. This field is a Required initial entry, and becomes read-only once the Path is defined.</td> </tr> <tr> <td rowspan="2">Leaf Item</td> <td>The Item at the endpoint of the Query Path. Target Properties are derived from this item. There can only be one Leaf Item in the query path.</td> </tr> <tr> <td>(Query Definition)</td> <td>Internally stored query definition describing (a) the Path from Root item to Leaf item and (b) Target Property on the Leaf item from which values will be derived.</td> </tr> <tr> <td><i>Read Only</i></td> <td>Target Property</td> <td>Name of the Property on the Leaf Item used to derive Attribute values. Type must match Datatype [2].</td> </tr> </tbody> </table> | “Applied To” Itemtype | The Itemtype that will define the scope of “CurrentItem” in Boolean expressions; also the Root Item of the Path defined by the <i>Embedded Query Definition</i> . This field is a Required initial entry, and becomes read-only once the Path is defined. | Leaf Item | The Item at the endpoint of the Query Path. Target Properties are derived from this item. There can only be one Leaf Item in the query path. | (Query Definition) | Internally stored query definition describing (a) the Path from Root item to Leaf item and (b) Target Property on the Leaf item from which values will be derived. | <i>Read Only</i> | Target Property | Name of the Property on the Leaf Item used to derive Attribute values. Type must match Datatype [2]. |
| “Applied To” Itemtype | The Itemtype that will define the scope of “CurrentItem” in Boolean expressions; also the Root Item of the Path defined by the <i>Embedded Query Definition</i> . This field is a Required initial entry, and becomes read-only once the Path is defined. | | | | | | | | | |
| Leaf Item | The Item at the endpoint of the Query Path. Target Properties are derived from this item. There can only be one Leaf Item in the query path. | | | | | | | | | |
| | (Query Definition) | Internally stored query definition describing (a) the Path from Root item to Leaf item and (b) Target Property on the Leaf item from which values will be derived. | | | | | | | | |
| <i>Read Only</i> | Target Property | Name of the Property on the Leaf Item used to derive Attribute values. Type must match Datatype [2]. | | | | | | | | |

The **Query Definition** is accessed by *double-clicking the Leaf Item column** of the Attribute Query. The Root item is pre-populated as the “Applied To” Itemtype. Using the standard QD user interface, define the Path from Root to Leaf Item, and also select the Target Property. On completion, the Read-Only columns “Leaf Item” and “Target Property” will be populated.



Using a derived attribute in a Boolean expression

Derived Multivalued Attributes are Collections, and the following operators support their use in Condition logic (Boolean expressions):

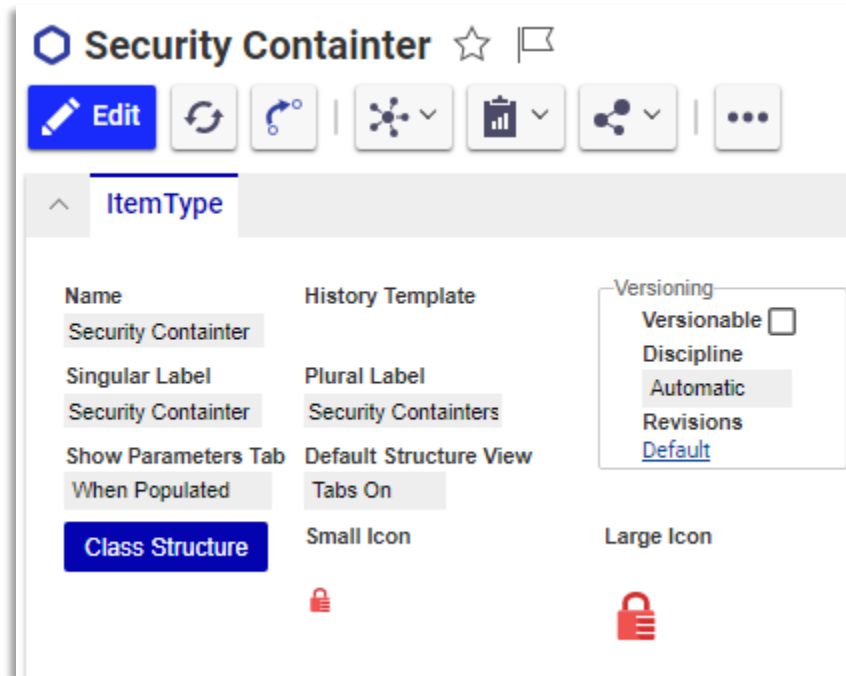
Collection Operators:

| Operator | Description |
|---|--|
| Collection.IsEmpty (<multival attribute>) | TRUE if Collection is empty |
| Collection.Contains (<multival attribute>, value) | TRUE if Collection <multival attribute> contains the value <value> (type |
| Collection.Overlaps (<multival attribute1>, <multival attribute2>) | TRUE if Collection <multival attribute1> has values in common with <multival attribute2> |

Recall that Derived Attributes are MAC Attributes that may be used in Boolean expressions. Because their type is Collection, the Operators are collection operators shown above.

Try Its: (Work along with the instructor through this section)

- 1) Create a 'container' item to easily apply MAC policies to Parts, Documents, etc.



The screenshot shows the 'Properties' window for the 'Security Containter' item. It features a toolbar with icons for adding, removing, searching, and a 'Hidden' dropdown menu. Below the toolbar is a table with the following data:

| Name | Label | Data Type | Data Source [...] | Le... |
|----------------|----------------|-----------|-------------------|-------|
| name | Name | String | | 64 |
| security_index | Security Index | Integer | | |

Properties RelationshipTypes Views Server Events Actions

ItemTypes

Hidden

| Tab O... | Relationship ... ↑ 1 | Tab Label | Name ↑ 2 |
|----------|----------------------|---------------|----------|
| 128 | Doc Security | Doc Security | Document |
| 256 | Part Security | Part Security | Part |
| 384 | User Security | User Security | User |

Security Containers

Search Clear Simple

| Security In... ↑ | Name |
|------------------|-------------------------|
| 1 | Controlled Unclassified |
| 2 | Public Trust Position |
| 3 | Confidential |
| 4 | Secret |
| 5 | Top Secret |
| 6 | Compartmented |

Next...

we'll create a Derived Attribute Definition that will allow Query results to be used in MAC Rules.
(Follow along with Instructor)

ContainerSecurity ☆ 🚩

Edit ↻ 🔁 | 🌐 ⌵ 📊 ⌵ 🌐 ⌵ | ⋮

Derived Attribute Definition

| Name | Datatype |
|-------------------|----------|
| ContainerSecurity | Integer |

Description

Get security level from container

Attribute Queries ☆

🗚️ 🗑️ | 🔍 🚫 Hidden ⌵ | 📷 ⌵ 🖥️ ⌵ 🌐 ⌵

| Applied To [...] | Leaf Item | Target Property |
|--------------------------|------------------------------------|-----------------|
| Document | Security_Container | security_index |
| Part | Security_Container | security_index |
| User | Security_Container | security_index |

And apply it in a MAC Policy against GET and Discover:

& || ! | ✓ ✕

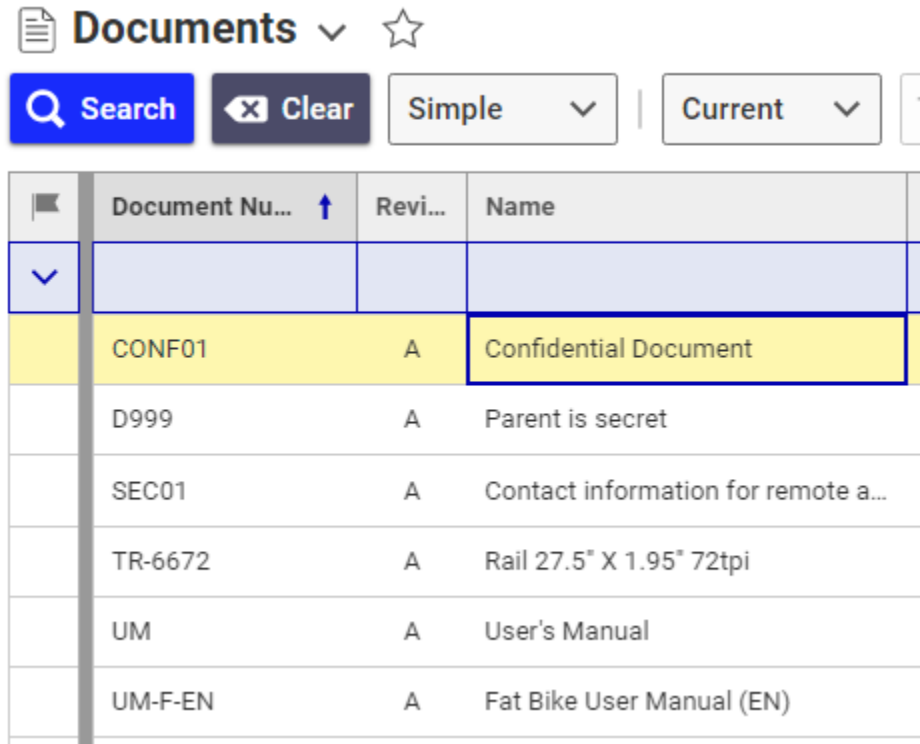
```
1 Collection.IsEmpty(CurrentItem.ContainerSecurity)
2 OR Collection.Overlaps(CurrentItem.ContainerSecurity, CurrentUser.ContainerSecurity)
```

(actual text)

Collection.IsEmpty(CurrentItem.ContainerSecurity)

OR Collection.Overlaps(CurrentItem.ContainerSecurity, CurrentUser.ContainerSecurity)

Finally, we will test various combinations of User / Part / Document Security Containment scenarios.



| Document Nu... ↑ | Revi... | Name |
|------------------|---------|-------------------------------------|
| CONF01 | A | Confidential Document |
| D999 | A | Parent is secret |
| SEC01 | A | Contact information for remote a... |
| TR-6672 | A | Rail 27.5" X 1.95" 72tpi |
| UM | A | User's Manual |
| UM-F-EN | A | Fat Bike User Manual (EN) |

Add Parts, Documents to various containers to see effects. Add Users to various containers to allow access.

Questions?