Overview

Aras Innovator provides the flexibility to give administrators many options when controlling permissions and access. Mandatory Access Control (MAC) is designed to provide access control to Items based on properties of the Item being requested, and properties of the user requesting the Item. MAC Policies can be created for several different use cases where a user's properties (Clearance Level, Citizenship, Location, etc..) determine access rights to Items which may also have various restricting properties (Classification, Ownership, Cost, etc.).

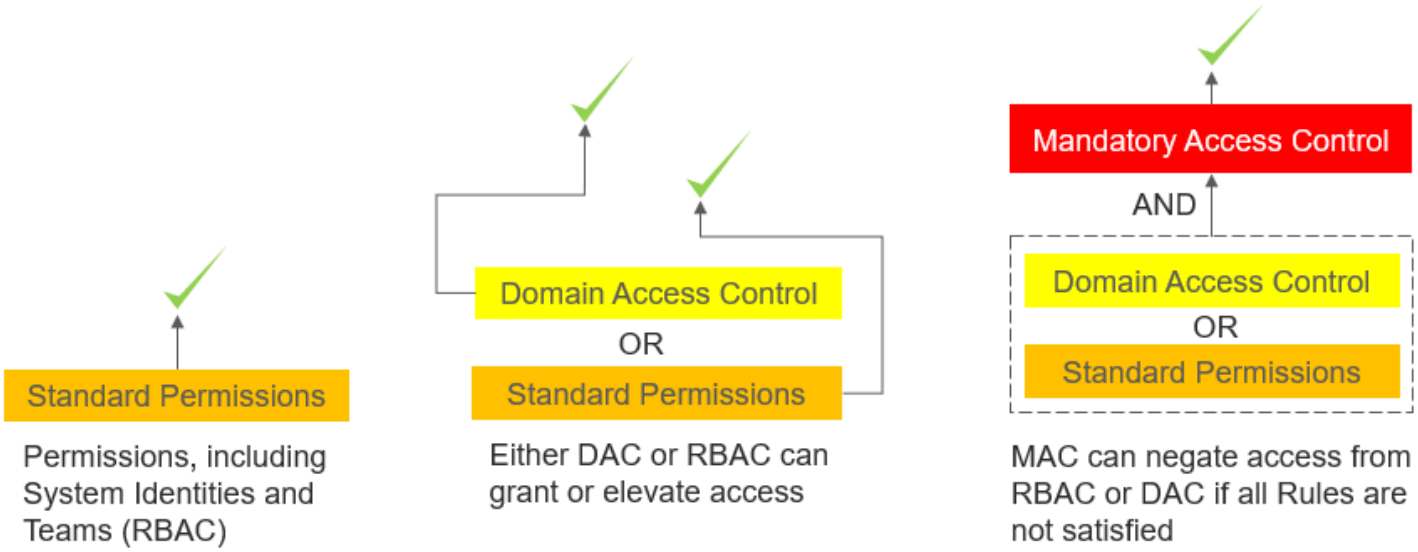
In addition to properties directly on Itemtypes and users, attributes can be derived from related items and used in conditional logic for the access control policy. For example, a policy may want to derive a list of security level restrictions from all referencing Projects for a Part.

Custom logic can also be used to determine environmental conditions, like time of day or other customer-specific requirements. All of these features empower administrators to implement a flexible and robust security scheme.

This overview provides a conceptual understanding of MAC Policy. Section 2 will guide you through the process of creating MAC Policies with step-by-step instructions.

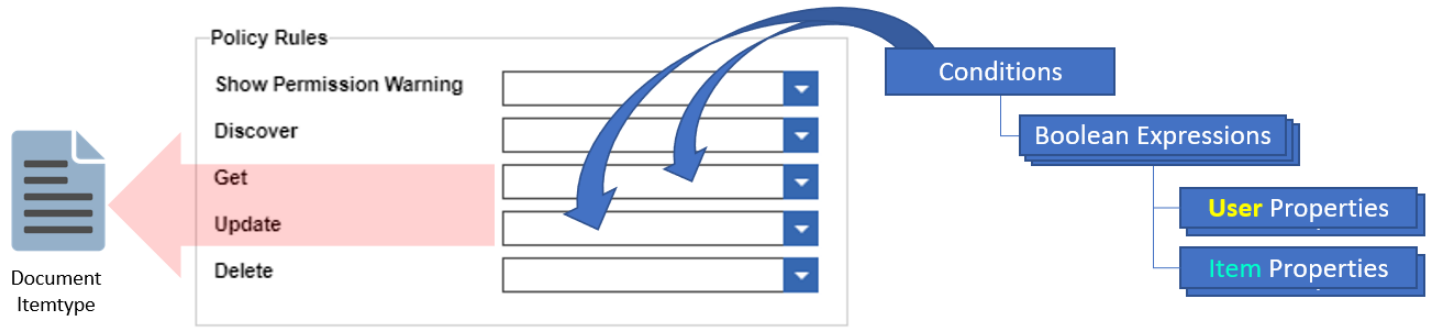
MAC Policy Concepts

Mandatory Access Control (MAC) imposes an additional layer of access control over Item permissions and Domain Access Control (DAC). MAC does not grant any access—it simply revokes or allows existing access by enforcement of a final and overriding set of conditions for access control using (a) the attributes of the item being accessed as well as (b) attributes of the User who is requesting access dynamically at the time the request is made.



How MAC Works

In its most basic usage MAC utilizes property values from the Item being accessed and the User requesting access and uses them in Condition logic to control access. Conditions are applied to Access Rights ('Get','Update','Delete', etc) to define a MAC Policy Rule. A set of MAC Policy Rules defines a MAC Policy, which can be applied to specified Itemtypes, e.g., *Document* in the example below.

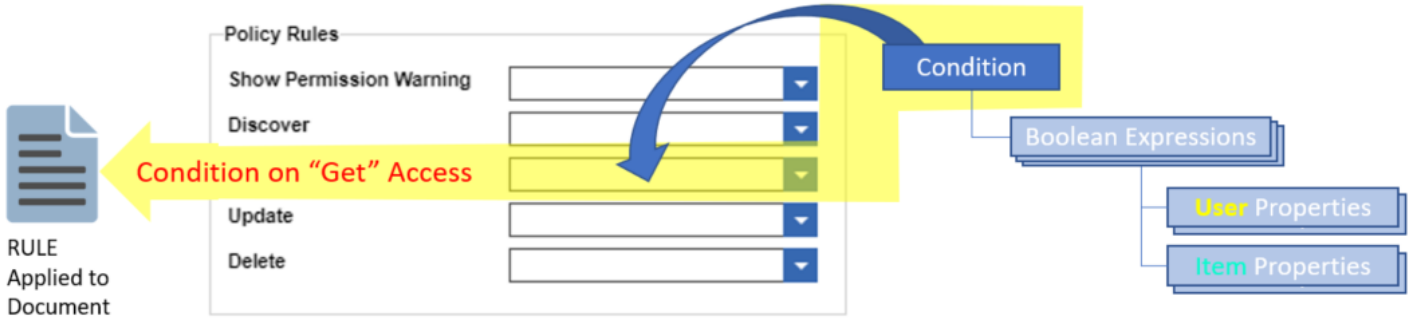


For example, the hypothetical Condition Logic for the above MAC controlled Document might contain the following boolean expressions:

1. Current **Item** (E.g., Document) has a property "requires security" set to TRUE
2. AND: The current **User** has a property "security clearance" with a value greater than 0
3. AND: The current **User** property "foreign national" is set to FALSE
4. AND: The Environment Attribute *Within_Accessible_Hours* is TRUE

If the result of the overall Condition is TRUE then the access rights it is associated with ('Get', 'Update') will remain unaffected for the requesting user. If the result of the Condition is NOT true, then MAC will unilaterally revoke any existing "Get" and "Update" access for the current user. In this manner, the MAC Policy effectively provides a final exclusionary filter over all other Access Controls that may be in place via RBAC and/or DAC.

MAC Policy Rules



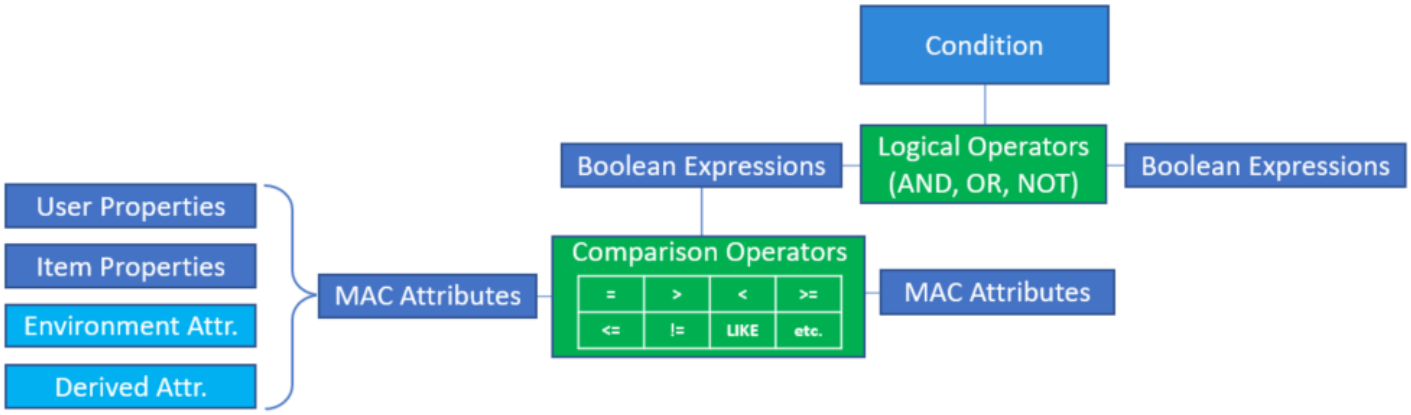
When a Condition is applied to an Access Right ("Get" in this example), then it defines a *MAC Policy Rule* for the Itemtype(s) that the MAC Policy applies to (Document in this example).

Configuring a Rule is a simple selection of an existing Condition item in the access type row. The actual logic behind the Rule is defined in the **Condition** Editor, and the Condition is defined in terms of the boolean expressions contained therein

MAC Conditions

Before getting into the step-by-step details of establishing a MAC Policy in Section 2 it's helpful to understand the hierarchy of Attributes, Boolean Expressions, and Operators that comprise a MAC Policy Condition.

Logically, the hierarchy looks like shown in Figure 4.



Boolean Expressions can be combined by Logical Operators AND, OR, NOT ultimately producing one TRUE or FALSE result as the Condition (parent). Recall the simple example from the Introduction Section 1.1 :

1. Current **Item** (E.g., Document) has a property “requires security” set to TRUE
2. AND: The current **User** has a property “security clearance” with a value greater than 0
3. AND: The current **User** property “foreign national” is set to FALSE
4. AND: The Environment Attribute *Within_Accessible_Hours* is TRUE

The result of evaluating all of these Boolean expressions as combined with Logical Operators is either TRUE or FALSE, therefore we can think of the Condition itself as having a boolean result for the Rule where it would be applied.

Boolean Expressions

Boolean expressions are the unit expressions that produce a TRUE/FALSE result using CurrentItem and CurrentUser objects. They are combined with Logical Operators (AND, OR NOT) and used to define Conditions.

For example, the above contains:

- Boolean expression 1 checking if User’s company name = ‘Aras’
- Boolean expression 2 seeing if the current item life cycle state is ‘Released’

- Overall Condition requiring that both (AND) are satisfied

Each boolean expression is defined as an evaluation of Attributes and constants via Comparison Operators, which result in TRUE or FALSE based on equality, greater or less than, equal to or greater/less than, LIKE (%) and inequality. Comparison Operators are covered in detail in Section 2.1.1.2 , special helper functions in Section 2.1.1.3 .

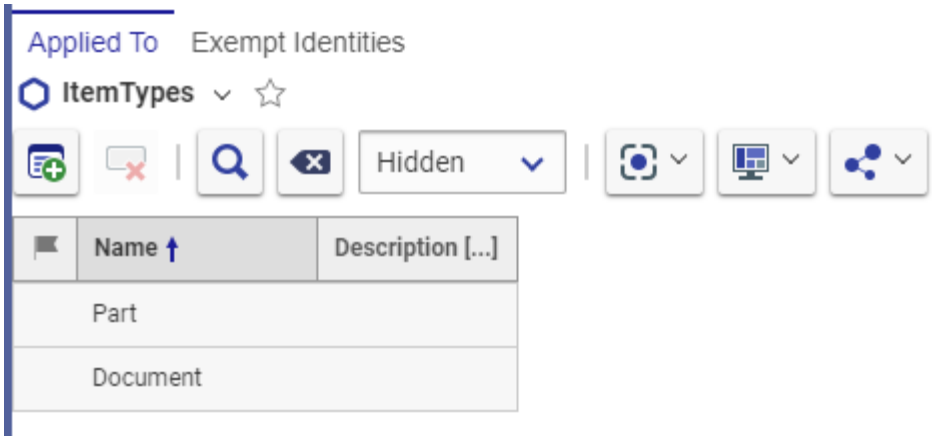
MAC Boolean expressions support xClass and xProperties, refer to Section 2.1.2 .

As previously mentioned, MAC Policy typically evaluates attributes associated with a User (requestor) vs. attributes on the (requested) Item to decide whether (or not) to revoke that User's access to the Item. Therefore, MAC Conditions most often involve dual expressions of the general form:

*<**User** attribute expression> AND <**Item** attribute expression>*

User attributes are referenced as ***CurrentUser**.<attribute>*, and Item attributes are accessed as ***CurrentItem**.<attribute>*:

- **CurrentUser** is always the current logged in user in the expression being evaluated in the Rule.
- **CurrentItem** is always the item being accessed, as configured in the MAC Policy under the "Applied To" tab.



Consider the example described in Table 2.

The first example of a conditional Permission

Applied To	Access Rights	Condition
Documents, Parts	Get, Discover	**CurrentUser**.clearance >= **CurrentItem**.security_level_required

The User has *Get and Discover* access to all Documents and Parts based on existing Permissions. However, if this Policy Rule was activated, then any user whose clearance level was less than the level required by the Document or Part being requested would have access rights revoked for that specific item instance upon attempt to access.

MAC does not necessarily require User or Item duality, but conditions of singularity can almost always be achieved using simple Groups or Teams and standard Permissions instead; see Table 3.

The second example of a conditional Permission

Applied To		Access Rights	Condition
Custom Secret	ItemType	Get, Discover	**CurrentUser**.clearance > 3

The **Custom Secret** ItemType can simply have a Permission assigned that allows members of the **Clearance3** group to have the **Get** and **Discover** Access Rights. Users can be moved in or out of the group by administrators without adding and maintaining properties on user instances, etc.

In summary, with very few exceptions MAC Policies employ Rules based on Conditions that juxtapose attributes of the current User against attributes of the current Item.

MAC Attributes

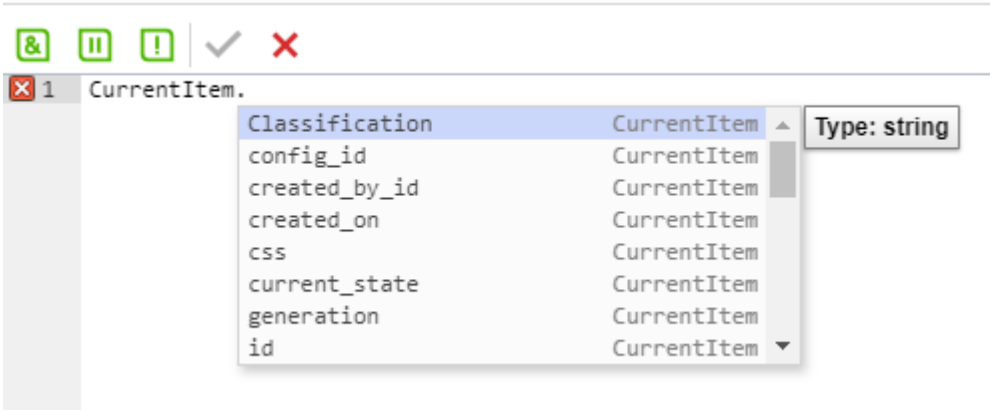
In MAC the term *attribute* is a general abstraction for any value or set of values that can be used within a boolean expression. They may be single-valued, or multi-valued, and used interchangeably where type and cardinality allow.

MAC Attributes may any one of the following:

1. A Constant of the correct datatype, cardinality, and value.
2. A Property defined on the current Itemtype being accessed.
3. A Property on the User Itemtype for the current user making the access request.
4. Environment Attribute dynamically set by execution of a (custom) Method.
5. Derived Multi-valued Attribute produced from the results of a Query Definition.

Enabling Properties for Use in MAC Expressions

Within a *CurrentItem* expression only those Properties that are added to container item *mp_PolicyAccessItem* will become available for use in MAC expressions. If you need additional properties to be available in the CurrentItem. `<attribute>` list shown here then the property should be added to this Item. Specific details are provided in Section 2.1.1.1 .



CurrentUser attributes are available as they are configured in the *User* itemtype. Only administrators should be allowed to edit the User item to prevent potential access change due to modification of properties. See Section 2.1.1 for details regarding this restriction.

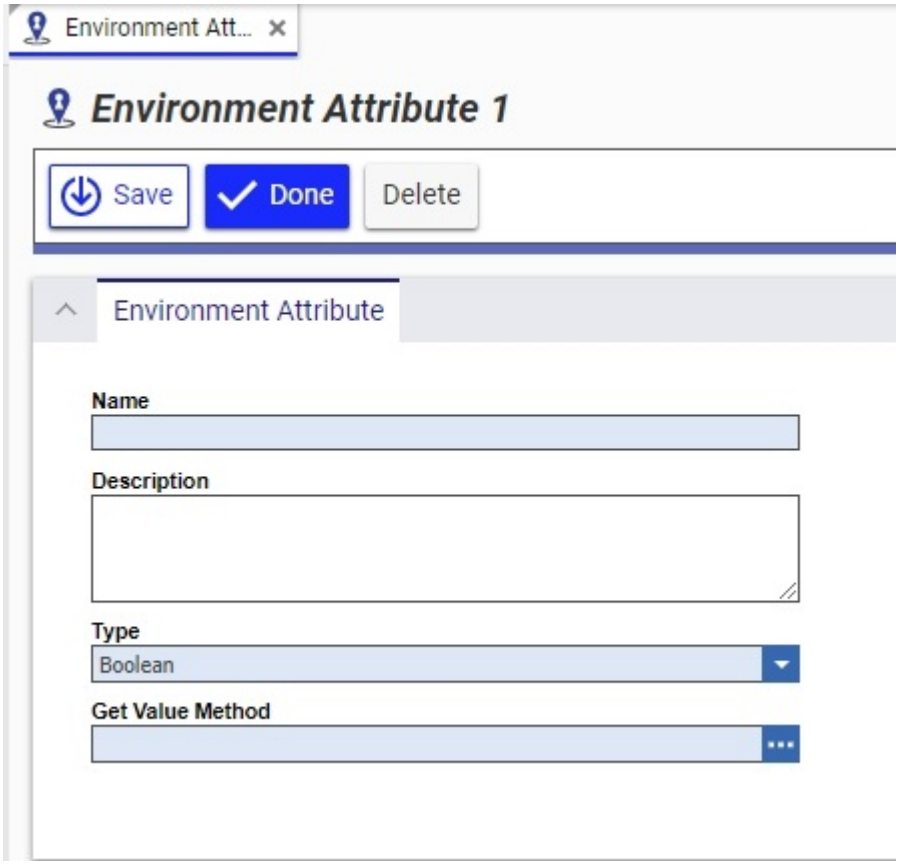
Environment Attributes

Environment attributes provide a way to customize specific access control circumstances such as geographical location, or the time an access request is being made – essentially any calculated attribute value for use in expressions. Recall once again the simple example from the Introduction Section 1.1 :

1. Current **Item** (E.g., Document) has a property **requires security** set to **TRUE**.
2. AND: The current **User** has a property **security clearance** with a value greater than 0.
3. AND: The current **User** property **foreign national** is set to **FALSE**.
4. AND: The **Environment** Attribute **Within_Accessible_Hours** is set to **TRUE**.

Within_Accessible_Hours (d) does not exist as a Property anywhere, it is a calculated ‘virtual’ attribute whose value is generated by execution of a Method.

MAC provides the **Environment Attribute** ItemType under **Administration** --> **Access Control** in the TOC with the UI shown in figure 8.



Note that a “Get Value Method” is required. It is the responsibility of the administrator to define and provide a Method as described in Section 2.2 .

Once this is completed, the named attribute becomes available in MAC Condition expressions.

Derived Multivalued Attributes

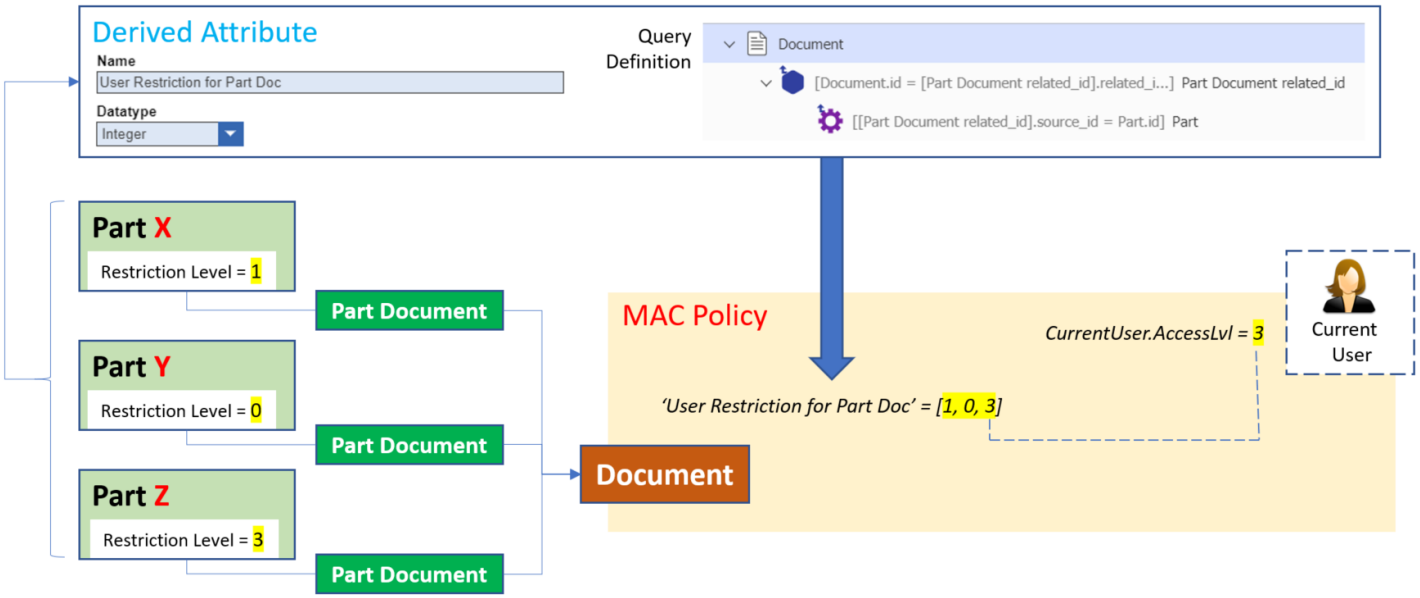
As of Innovator 12.0 SP6, access administrators may take advantage of a new capability in MAC to use the results of a Query Definition to generate a *derived multivalued* attribute for use in their MAC policy’s boolean expressions.

In the Query Definition shown in the following diagram, *Document* is the root item for the query definition, and *Part Document* reference (relationship) defines a path to a *Part* Itemtype parent. When this query executes, all of the “Restriction Level” target property values from all parent Parts are

derived into a multivalued attribute as a collection of integer values [1, 0, 3]. This collection can be referenced in boolean expressions by name.

Important

Any time spaces are used in an attribute name use [square brackets] in the expression to refer to it.



For the sake of simplicity *CurrentUser.AccessLvl* is a single valued attribute i.e., an integer property called 'AccessLvl' on the User Itemtype in the example above. Alternatively, a derived multi-valued attribute or a constant value could be used on the User Item as needed. In any case, these attributes can be used in a MAC Rule's condition expression as follows:

```
1  Collection.Contains(CurrentItem.[User Restriction for Part Doc],CurrentUser.AccessLvl)
```

The above Condition would only be true if the derived attribute collection for the current item *contains* the access level value from the current user, otherwise it is false.

Derived Attributes vs. Single-valued Attributes

Derived attributes provide greater flexibility over single-valued attributes for a number of reasons. It is not necessary to have the attribute configured as a Property on the *CurrentUser or CurrentItem* if the property can be ‘derived’ from a related item whether a parent or child relationship. Thus, it is not required to modify the User (Core) Itemtype if related to the parent item containing the attribute.

As discussed previously, Derived attributes are multi-valued. MAC boolean expressions can evaluate *multivalued* derived attributes as ‘Collections’ using set-based operators (Contains, Overlaps, IsEmpty).

Important

Property-based (single-valued) Attributes may result in better performance vs. Derived Attributes, which will provide greater flexibility. This should be taken into consideration when designing and testing a MAC Policy implementation.