Example 3: Bronze and its member group, Gold, have conflicting entitlements.

- Entitlement 1: Deny Bronze Group access to index.html
- Entitlement 2: Allow Gold Group access to index.html

This must be resolved by the policy conflict resolution setting. Therefore, attempts to gain access produce the following results:

- User 2 is allowed access because Gold group is more specific than Bronze group.
- User 1, however, is a member of both groups. Therefore, the groups are of equal specificity to the user. In this case, the system considers the policy conflict resolution setting specified for the resource index.html.

Policy Conflict Resolution

Policies that allow access, or policies that deny access, take priority depending on the policy conflict resolution setting of the resource in question.

For more information, see "Policy Conflict Resolution" on page 49.

Smart Rules

You use Smart Rules to allow or deny a user access to a resource based on the value of a user property at the moment the user attempts to access the resource. You can apply Smart Rules to any resource.

When a user tries to access a protected resource, Access Manager checks the user properties associated with the Smart Rules protecting the resource and grants or denies access based on the criterion of each Smart Rule.

Important: Smart Rules decide a user's access to a specified application only if no relevant entitlement exists at any level (user or group). Entitlements always take precedence over Smart Rules.

Each Smart Rule compares a specific user property value to an administrator-specified comparison criterion according to one of the comparison operators in the following table.

Data Type	Operator
Date	Before, After, Is Equal
Boolean	Is
String	Starts With, Contains, Does Not Contain, Ends With, Is Equal To, Is Greater Than, Is Greater Than Or Equal To, Is Less Than, Is Less Than Or Equal To, Is Not Equal To
Integer, Float	>=, <, =, >, <=, !=

For example, an online banking company can create a Smart Rule that allows only users with an account balance of more than $500.00 to access a certain page on their web site. This Smart Rule would take the following form:

Allow if Account Balance > 500

In this case, the user property value is "Account Balance." The comparison criterion is "500." The comparison operator is "greater than" (>).

If users have an account balance of more than $500.00, they are allowed access to the page. If users have an account balance of less than $500.00, they are denied access to the page. When a user is denied access to a resource, the system presents an HTTP 404 "File Not Found" message.

For information about creating a Smart Rule, see the Help topic "Adding Smart Rules."

Types of Smart Rules

There are three types of Smart Rules: Allow, Deny, and Require. These three types can be combined in various ways to implement business rules in controlling access to a resource.

If multiple Smart Rules protecting the same resource contradict each other, a policy conflict occurs. For information about how Access Manager resolves policy conflicts, see "Policy Conflict Resolution" on page 49.

Allow Rules

If a user property matches an Allow rule, the user is allowed access. For example, Resource A is protected by this Smart Rule: Allow if State Equals CA.

User A has the State property set with the value "CA." User B has the State property set with the value "WA."

User A's state property matches this Allow rule, so User A is allowed access. User B's state property does not match this Allow rule, so User B is not allowed access to Resource A.

If a resource is protected by several Allow rules, access is allowed if a user property matches any rule.

Deny Rules

If a user property matches a Deny rule, the user is denied access. For example, Resource A is protected by this Smart Rule: Deny if Age < 21.

User A has the Age property set with the value "18." User B has The Age property set with the value "30."

User A's age property matches the Deny rule, so User A is denied access to Resource A. User B's age property does not match the deny rule, so User B is allowed access to Resource A.

If a resource is protected by several Deny rules, access is denied if a user property matches any rule.

Require Rules

If a user property matches a Require rule, the user is allowed access. For example, Resource A is protected by this Smart Rule: Require Account > 500.

User A has the Account property set with the value "600." User B has the Acccount property set with the value "400."

User A's account property matches the Require rule, so User A is given access to Resource A. User B's account property does not match the Require rule, so User B is not allowed access to Resource A.

If a resource is protected by several Require rules, access is granted only if a user's properties match all the rules.

Deny Rules and Authorization Modes

If a user has a property with a value "N/A" (not yet entered), a Deny rule based on that property does not deny access for that user. If that user's access status is not established by a higher level entitlement for that resource, access status is determined by one of these settings:

- The Authorization Server set to active mode allows access.
- The Authorization Server set to passive mode denies access.

For more information about Deny rules, see "Deny Rules" on page 55.

For more information about active and passive modes, see "Authorization Mode: Active and Passive Modes" on page 39.

Combining Smart Rules

Allow rules and Deny rules can be combined with Require rules to implement business rules when controlling access to a resource. If Smart Rules of different kinds all protect a resource, the order in which they are evaluated is governed by the policy conflict resolution setting applied to the resource. For more information, see "Policy Conflict Resolution" on page 49.

The policy resolution setting is selected during the process of assigning a resource to be protected by Access Manager. When the default conflict resolution setting, "Allow access when policy conflicts occur," is selected, Allow rules are evaluated first, then Deny rules, and finally Require rules. When the alternative setting, "Deny access if policy conflicts occur," is selected, Deny rules are evaluated first, then Allow rules, and finally Require rules. The Smart Rules are evaluated until a user is either denied or allowed access.

The following examples show how four users with different properties are evaluated when trying to access a resource protected by an Allow, a Deny, and a Require rule.

Example 1

The following table shows the Smart Rules protecting Resource A and the properties of User A.

Smart Rules Protecting Resource A	User A Properties
Allow if State Equals CA	State with the value TX
Deny if Age < 21	Age with the value 23
Require Valid Credit Card Is True	Valid Credit Card with the value True

If the policy conflict resolution for this resource is set to "Allow access when policy conflicts occur," the Smart Rules are evaluated in this order:

1. Allow if State Equals CA
2. Deny if Age < 21
3. Require Valid Credit Card Is True

In this case, User A's State property (TX) does not match the "Allow if State Equals CA" rule, so User A is denied access to Resource A. Because the Allow rule has denied access, the remaining Smart Rules are not evaluated. By eliminating unnecessary evaluation of rules, system performance is improved.

Example 2

The following table shows the same Smart Rules protecting Resource A and the properties of User B.

Smart Rules Protecting Resource A	User B Properties
Allow if State Equals CA	State with the value CA
Deny if Age < 21	Age with the value 18
Require Valid Credit Card Is True	Valid Credit Card with the value True

If the Policy Conflict Resolution for this resource is set to "Deny access when policy conflicts occur," the Smart Rules are evaluated in this order:

1. Deny if Age < 21
2. Allow if State Equals CA
3. Require Valid Credit Card Is True

In this case, User B's Age property (18) matches the "Deny if Age < 21" rule, so User B is denied access to Resource A. Because the Deny rule has denied access, the remaining Smart Rules are not evaluated.

Example 3

The following table shows the same Smart Rules protecting Resource A and the properties of User C.

Smart Rules Protecting Resource A	User C Properties
Allow if State Equals CA	State with the value CA
Deny if Age < 21	Age with the value 23
Require Valid Credit Card Is True	Valid Credit Card with the value False

If the Policy Conflict Resolution for this resource is set to "Deny access when policy conflicts occur," the Smart Rules are evaluated in this order:

1. Deny if Age < 21
2. Allow if State Equals CA
3. Require Valid Credit Card Is True

In this case, User C does not match the "Deny if Age < 21" rule, matches the "Allow if State Equals CA" rule, but does not match the "Require Valid Credit Card Is True" rule, so User C is denied access to Resource A.

Example 4

The following table shows the same Smart Rules protecting Resource A and the properties of User D.

Smart Rules Protecting Resource A	User D Properties
Allow if State Equals CA	State with the value CA
Deny if Age < 21	Age with the value 23
Require Valid Credit Card Is True	Valid Credit Card with the value True

If the Policy Conflict Resolution for this resource is set to "Deny access when policy conflicts occur," the Smart Rules are evaluated in this order:

1. Deny if Age < 21
2. Allow if State Equals CA
3. Require Valid Credit Card Is True

In this case, User D does not match the "Deny if Age < 21" rule, matches the "Allow if State Equals CA" rule, and matches the "Require Valid Credit Card Is True" rule, so User D is given access to Resource A.

Evaluating Smart Rules in Sequential Order

Access Manager can be configured to ignore policy conflict resolution settings and evaluate the Smart Rules in the order they appear in the Administrative Console. For more information, see your system administrator.

By configuring Access Manager to ignore policy conflict resolution settings, it is possible to arrange different types of Smart Rules in an arbitrary list. The following table shows a group of Smart Rules that control access to the wine section of an online supermarket as they appear in the Administrative Console:

Smart Rule	Access
Age >= 21	Require
Valid Credit Card Is True	Require
Encryption Off Is True	Deny
Bad Credit Is False	Require
Account Closed Is True	Deny
Valid Username Is True	Allow
Valid PIN Is True	Allow

Because the "Age >= 21" Require rule must be matched in order to shop in the wine section, it is the most important rule in this group. It is placed at the top of the list so that the other Smart Rules do not need to be evaluated if the first Rule is not matched. Reducing unnecessary evaluation of rules improves performance.

Smart Rules and User Properties with Multiple Values

Smart Rules can include user properties that have more than one value.

For example, a resource is protected by this Smart Rule: Allow if Department Equals Sales.

User A works in more than one department and has the Department property set with the values "Marketing," "Sales." User B also works in more than one department and has the Department property set with the values "Marketing," "Customer Support."

In this case, User A is allowed access to this resource because the Allow rule is matched (User A is in Sales). User B, however, is denied access because the Allow rule is not matched (User B is not in Sales).

Smart Rules with multi-value user properties that use the operators "Does Not Contain" or "Does Not Equal" are only satisfied if none of the property values match the comparison criterion. For all other operators, only one of the property values needs to match the comparison criterion to satisfy the Smart Rule.

Smart Rule Examples

These additional examples further illustrate the use of Smart Rules.

Example One

An insurance company using Access Manager has customers throughout the south and northwest United States. At the beginning of its fiscal year, the company decided to make a special offer available to residents of California, Texas, and Oregon on its web site. To accomplish this, the company created three Smart Rules to control access to the page on the web server containing the special offer. The company had already created a property called State, which is normally used as part of the customer's mailing address:

- Allow if State Equals CA
- Allow if State Equals TX
- Allow if State Equals OR

This simple setup accomplished the desired goal. Residents of California, Texas, and Oregon can access the special offer, but everyone else is denied access.

A month later, the insurance company decided to limit the offer to users with good credit ratings. Since there is already another property called Bad-Credit, (a Boolean that is set to true if the account has been flagged for non-payment), adding another Smart Rule is straightforward:

- Deny if Bad Credit Is True
- Allow if State Equals CA
- Allow if State Equals TX
- Allow if State Equals OR

Because the Policy Conflict Resolution setting for this resource is set to, **"**Deny access when policy conflicts occur," Deny rules are evaluated first. Only users with good credit from California, Texas, or Oregon can access the insurance company's special offer web page.

Example Two

It is also possible to combine Require rules. In this example, a company wants to limit access to an area of its web site to retail customers that have account balances over $100. In this case, both parts of the condition must be met or the user is denied access. The Access Manager Administrator creates two Smart Rules:

- Require Account Balance > 100
- Require Account Type Equals Retail

At runtime, only Retail users with account balances in excess of $100 are allowed access to the site.