

Resource Name Condition Rewrite Guide

Overview

Use of condition attributes on unsupported permissions in your IAM policies will begin blocking access. To avoid any surprising effects from this change, your currently configured condition expressions may need to be rewritten.

This document provides rewriting guidance on resource name condition expressions with representative examples and suggested actions.

Supported Service Permissions for Resource Name Condition

See [resource.name](#) for details on the behavior of policy evaluation when an unsupported permission is used with the Resource Name condition.

Specifically, see [Resource Name Format](#) for the detailed list of resource types and their permissions in scope of supporting the Resource Name condition. Permissions not supported, but used in a binding with Resource Name condition, will not be granted.

Valid Resource Name

To check if the string used with resource.name in a condition expression is valid, see [Resource Name Format](#) for details. In particular, please check on two factors:

- Only resource types and resource name format syntax enumerated in the document are in scope for Conditions support.
- For the intended usage for a resource type, only the enumerated resource name format in the document is considered valid.

Example Cases

The following are conditional role binding cases that relate to usage of the resource.name attribute, primarily focusing on negation of resource name (!= or !...startsWith()).

Case 1: no permission in the role supports resource name, and disjunction

Example Expression	<code>resource.name != <string literal> <other subexpression> !resource.name.startsWith(<prefix string>) <other subexpression></code>
Example Role	<code>roles/iam.roleViewer</code>
Description	No supported permission in the associated role for resource name condition.

	This portion of the subexpression will be evaluated to FALSE. Hence the rest of <other subexpression> can cover the full functionality for this condition expression.
Suggested actions	1) Remove the resource.name related subexpression, or 2) Remove the condition if <other subexpression> will not take effect as expected if resource name subexpression is removed The choice between #1 and #2 depends on the intended usage of this binding.

Case 2: no permission in the role supports resource name, and conjunction

Example Expression	<code>resource.name != <string literal> && <other subexpression> !resource.name.startsWith(<prefix string>) && <other subexpression></code>
Example Role	<code>roles/iam.roleViewer</code>
Description	No supported permission in the associated role for resource name condition. This portion of the subexpression will be evaluated to FALSE. Hence, the entire condition expression will be evaluated to FALSE.
Suggested actions	1) Remove the resource.name related sub expression to keep the effect by the rest of expression, or 2) Remove the condition if <other subexpression> will not take effect as expected if resource name subexpression is removed, making the binding unconditional, or 3) Remove the binding completely The choice between #1, #2, and #3 depends on the intended usage of this binding.

Case 3: invalid resource name strings, and mixed permissions

Example Expression	<code>resource.name != "foobar"</code> or <code>! resource.name.startsWith("foobar")</code>
Example Role	<code>roles/compute.Admin</code>
Description	The string literal used in the expression is invalid. The role has mixed of supported and unsupported permissions for resource.name condition

	Condition evaluation on this expression will change behavior: - Before: effective TRUE for all permissions - After: TRUE for support permissions; FALSE for unsupported permissions
Suggested actions	1) Fix the existing condition a) Fix resource name string format according to the intended resource type and permission. Reference: https://cloud.google.com/iam/docs/conditions-resource-attributes#resource-name. b) And, add resource type guard to ensure the resource name restriction is forced on the intended permissions, and leaving other permissions as granted unconditionally. See https://cloud.google.com/iam/docs/configuring-resource-based-access for guidance on adding type guard to your resource name condition. 2) Remove this condition or subexpression, to keep the effective TRUE behavior, or 3) Remove this conditional binding The choice between #1, #2, and #3 depends on the intended usage of this binding.
Example Updated Expression	<pre>(resource.type == 'compute.googleapis.com/Disk' && resource.name.startsWith('projects/project-123/regions/us-central1/disks/staging')) resource.type != 'compute.googleapis.com/Disk'</pre>

Case 4: valid resource name strings and mixed permissions

Example Expression	<code>! resource.name.startsWith("projects/_/buckets/secret-foobar")</code>
Example Role	<code>roles/storage.admin</code>
Description	The string literal used in the expression is valid The role has mixed of supported and unsupported permissions for resource.name condition Condition evaluation on this expression will change behavior: - Before: effective TRUE for all permissions

	- After: TRUE for support permissions; FALSE for unsupported permissions
Suggested actions	Fix the existing condition expression by adding resource type/service guard to ensure the resource name restriction is forced on the intended permissions, and leaving other permissions as granted unconditionally. See https://cloud.google.com/iam/docs/configuring-resource-based-access for guidance on adding type guard to your resource name condition.
Example Updated Expression	<pre>(! resource.name.startsWith("projects/_/buckets/secret-foobar")) (resource.type != 'storage.googleapis.com/Bucket' && resource.type != 'storage.googleapis.com/Object')</pre>

Case 5: subset of supported permissions relevant to the resource name string

Example Expression	<code>!resource.name.startsWith("projects/my-project/regions/australia-southeast1/disks/foobar-disk")</code>
Example Role	<code>roles/compute.storageAdmin</code>
Description	- The string literal used in the expression is valid - The string literal implies the restriction on certain resource type, hence out of all supported permissions in the role, only subset are relevant to this resource type - In this example, the string is for Compute Disks. - In addition, there may be unsupported permissions in the role The intent may be: resource name restriction on Compute Disk, no restriction on other compute resource permissions.
Suggested actions	Adding a type guard can make it explicit that the intended scope is for the relevant resource type only, and other permissions are granted unconditionally. See https://cloud.google.com/iam/help/conditions/resource-name-scope for guidance on adding type guard to your resource name condition.
Example Updated Expression	<code>!resource.name.startsWith("projects/my-project/regions/australia-southeast1/disks/foobar-disk") resource.type != 'compute.googleapis.com/Disk'</code>

Case 6: valid resource name with custom role

Example Expression	<code>!resource.name.startsWith("projects/my-project/regions/australia-so</code>
---------------------------	--

	<code>utheast1/disks/foobar-disk")</code>
Example Role	<code>projects/my-project/roles/some-role</code>
Description	- Unclear if the custom role contains supported permissions or not - The string literal used in the expression is valid - The string literal implies the restriction on certain resource type, hence out of all the supported permissions in the role, only the subset that are relevant to this resource type - In this example, the string is for Compute Disks. The intent may be: resource name restriction on Compute Disk, no restriction on other compute resource permissions.
Suggested actions	- Remove condition, If the custom role contains no supported permissions - Adding a type guard, if custom role contains compute Disk permissions. See case 4.
Example Updated Expression	<code>!resource.name.startsWith("projects/my-project/regions/australia-southeast1/disks/foobar-disk") resource.type != 'compute.googleapis.com/Disk'</code>

Case 7 (optional): valid resource name and type guard, but too restrictive

Example Expression	<code>(resource.type == "storage.googleapis.com/Bucket" && !resource.name.startsWith("projects/_/buckets/foobar")) (resource.type == "storage.googleapis.com/Object" && !resource.name.startsWith("projects/_/buckets/foobar"))</code>
Example Role	<code>roles/storage.admin</code>
Description	This case is optional to fix. The condition expression clearly adds the appropriate restriction on target resource types, with resource name condition. The only potential issue: the role contains a mix of supported and unsupported permissions. The unsupported permissions are explicitly not granted by the condition. In the above example, <code>resourcemanager.projects.get/list</code> permissions are not allowed, because of the usage of type guards.
Suggested actions	- Do nothing, if restricting to storage permissions are intended, or - Add additional type guard to grant <code>resourcemanager.projects.*</code> permissions without further restriction.

	Note: the following expression can be further simplified
Example Updated Expression	<pre>(resource.type == "storage.googleapis.com/Bucket" && !resource.name.startsWith("projects/_/buckets/foobar")) (resource.type == "storage.googleapis.com/Object" && !resource.name.startsWith("projects/_/buckets/foobar")) (resource.type != "storage.googleapis.com/Bucket" && resource.type != "storage.googleapis.com/Object")</pre> Or <pre>!resource.name.startsWith("projects/_/buckets/foobar") (resource.type != "storage.googleapis.com/Bucket" && resource.type != "storage.googleapis.com/Object")</pre>

Case 8 (optional): valid resource name and type guard, MAY be too restrictive

Example Expression	<pre>resource.type == 'compute.googleapis.com/Instance' && ! resource.name.extract('/instances/{name}').startsWith('foobar')</pre>
Example Role	Some custom role
Description	This case is optional to fix. Very similar to Case 7, except the role used is a custom role. There maybe two issues: 1. The same issue as described in Case 7, or 2. The custom role contains no supported permissions
Suggested actions	- Remove condition, if custom role contains no compute instance permission, or - Do nothing, if restricting to storage permissions are intended, or - Add additional type guard to grant resourcename.projects.* permissions without further restriction. Note: the following expression can be further simplified
Example Updated Expression	<pre>(resource.type == 'compute.googleapis.com/Instance' && ! resource.name.extract('/instances/{name}').startsWith('foobar')) resource.type != 'compute.googleapis.com/Instance'</pre> Or <pre>resource.name.extract('/instances/{name}').startsWith('foobar') resource.type != 'compute.googleapis.com/Instance'</pre>