Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 May 2026

P. Poreddy
Individual Contributor

November 2025

SCIM RoleAssignment Draft Specification v0.2
draft-poreddy-scim-role-assignment-01

Abstract

SCIM 2.0 defines "roles" and "entitlements" attributes on the User resource, but it lacks a standardized way to bind roles to specific scopes such as projects, tenants, or groups. This gap forces organizations to rely on group sprawl or non-standard encodings, preventing true interoperability. This document introduces a new SCIM resource type, RoleAssignment, which models scoped role bindings as first-class records, enabling portable provisioning, lifecycle governance, and compliance visibility.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 May 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

The SCIM protocol [RFC7643] [RFC7644] defines the User and Group resources and allows global roles to be attached to Users via the "roles" attribute. However, the specification does not provide a standardized way to associate roles with specific scopes such as projects, tenants, or device groups.

This limitation prevents SCIM from modeling the most common real-world requirement: assigning different roles to the same identity in different contexts.

For example, consider a user named Alice:

```
User: Alice
 +-- Global Role: Power User
 +-- Project A: Maintainer
 +-- Project B: Developer
 +-- Project C: ReadOnly
```

Today, there is no interoperable SCIM method to represent Alice’s per-project role bindings. Current workarounds include creating Groups for every {scope x role} combination, which leads to group sprawl and poor interoperability, or embedding scope names into free-form role strings, which are not machine-readable or portable.

These limitations are visible in real-world SCIM implementations: GitLab [GITLAB-SCIM], Tanium [TANIUM-RBAC], and scenarios in Microsoft Entra ID [AZURE-SCIM].

1.2. Proposed Solution

This document introduces a new SCIM 2.0 resource, _RoleAssignment_, which makes scoped role bindings a first-class concept. Each RoleAssignment explicitly links a subject (e.g., User), a scope (e.g., Project), and a role (e.g., Developer). Optional metadata such as validity periods, source system, and approver information enable lifecycle management and governance.

By standardizing RoleAssignments:

* Identity Providers can provision scoped roles in a portable way.
* Service Providers can expose and consume these assignments consistently.
* Auditors and governance systems can query "who has what role in which scope."

1.3. Design Rationale

This specification introduces RoleAssignment as a standalone resource type rather than extending existing SCIM resources for the following reasons:

1.3.1. Why not extend User.roles?

User.roles is a multi-valued attribute without scope information. Extending it would require complex nested structures that don’t fit SCIM’s flat attribute model and would make querying and lifecycle management difficult.

1.3.2. Why not use Groups?

Using Groups for every {scope × role} combination leads to group sprawl and poor scalability. A system with 100 projects and 5 roles would require 500 groups. Group membership also lacks the metadata needed for governance (validity periods, approval chains).

1.3.3. Why a connection resource?

Unlike SCIM’s traditional approach of embedding relationships, RoleAssignments have their own lifecycle, metadata, and audit requirements. They need: - Independent querying ("show all Developer roles in Project X") - Temporal validity and expiration - Approval and provenance tracking - Status transitions independent of subject or scope

1.3.4. Why immutable core attributes?

Making subject, scope, and role immutable ensures audit integrity. Each RoleAssignment represents a discrete authorization event. Changes to these core attributes represent new authorization decisions and should be tracked as separate resources.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Overview

The RoleAssignment resource complements existing SCIM resources. Whereas the User.roles attribute provides only coarse global roles, RoleAssignment expresses who (subject) has what role in which scope. This allows interoperable provisioning of scoped role bindings.

4. Schema

4.1. Resource Type

```
{
  "name": "RoleAssignment",
  "endpoint": "/RoleAssignments",
  "schema": "urn:ietf:params:scim:schemas:core:2.0:RoleAssignment",
  "schemaExtensions": []
}
```

4.2. Attributes (Top-Level)

The RoleAssignment resource defines the following top-level attributes.

Attribute	Type	Req	Multi	Mutability	Description
subject	complex	yes	no	immutable	The subject receiving the role.
scope	complex	yes	no	immutable	The scope where the role applies.
role	complex	yes	no	immutable	The role granted within the scope.
priority	integer	no	no	readWrite	Conflict resolution (higher wins). Default 0.
grant	complex	no	no	readWrite	Assignment metadata (source, reason, approver).
validity	complex	no	no	readWrite	Validity window in UTC.
status	string	no	no	readOnly	Computed lifecycle status.

Table 1

"The common attributes (id, externalId, meta) defined in RFC 7643 Section 3.1 are inherited by RoleAssignment and are not redefined here." _Priority usage guidance:_ When multiple active assignments exist for the same subject and scope, the assignment with the highest priority takes precedence. If priorities are equal, implementation-defined resolution rules apply.

4.3. subject Sub-Attributes

Attribute	Type	Req	Multi	Mutability	Description
value	string	yes	no	immutable	Identifier of the subject. If the subject is a resource, value MUST equal the id attribute of that resource.
$ref	reference	no	no	immutable	URI to the subject resource, if the subject is a resource.
type	string	no	no	immutable	Subject type. If the subject is a resource, type MUST equal the resource type name of that resource.
display	string	no	no	immutable	A human-readable name for the subject.

Table 2

4.4. scope Sub-Attributes

Attribute	Type	Req	Multi	Mutability	Description
type	string	yes	no	immutable	Scope type. Common: "project", "tenant", "organization", "application", "environment".

value	string	yes	no	immutable	Provider-meaningful identifier of the scope.
$ref	reference	no	no	immutable	URI to the scope resource, if available. Optional when scope is opaque.
display	string	no	no	immutable	A human-readable name for the scope.

Table 3

4.5. role Sub-Attributes

Attribute	Type	Req	Multi	Mutability	Description
value	string	yes	no	immutable	Stable identifier for the role. If the role is a resource, value MUST equal the id attribute of that resource.
display	string	no	no	immutable	Human-readable name of the role.
$ref	reference	no	no	immutable	URI to a role catalog entry or role resource, if the role is a resource.
type	string	no	no	immutable	Role type. If the role is a resource, type MUST equal the resource type name of that

					resource.

Table 4

Role Discovery Guidance: Since role.$ref is optional, servers SHOULD support filtering on role and scope to enable discovery, for example:

```
GET /RoleAssignments?attributes=role,scope&
    filter=scope.type eq "project"
```

Providers that expose a role catalog MAY align discovery with the SCIM Roles and Entitlements approach [SCIM-ROLES-ENTITLEMENTS].

4.6. grant Sub-Attributes

Attribute	Type	Req	Multi	Mutability	Description
source	string	no	no	immutable	Originating system or process.
reason	string	no	no	readWrite	Human-readable justification.
approver	complex	no	no	immutable	The approver of this assignment.

Table 5

4.6.1. approver Sub-Attributes

Attribute	Type	Req	Multi	Mutability	Description
value	string	yes	no	immutable	Identifier of the approver. If the approver is a resource, value MUST equal the id attribute of that resource.

$ref	reference	no	no	immutable	URI to the approver resource, if the approver is a resource.
type	string	no	no	immutable	Approver type. If the approver is a resource, type MUST equal the resource type name of that resource.
display	string	no	no	immutable	A human-readable name for the approver.

Table 6

4.7. validity Sub-Attributes

Attribute	Type	Req	Multi	Mutability	Description
validFrom	dateTime	no	no	readWrite	Start of validity window ([RFC3339] format with optional timezone offset).
validTo	dateTime	no	no	readWrite	End of validity window ([RFC3339] format with optional timezone offset).

Table 7

4.8. Resource Mutability

RoleAssignment resources have restricted mutability to maintain audit integrity:

IMMUTABLE attributes (cannot be changed after creation): - subject, scope, role: The core binding cannot change - grant.source, grant.approver: Provenance must be preserved

MUTABLE attributes (can be updated via PATCH/PUT): - priority: Allow conflict resolution - validity: Allow time-bound extensions - grant.reason: Allow justification updates

To change immutable attributes, clients MUST delete the existing assignment and create a new one. This ensures each role grant is an explicit, auditable event.

4.9. status Semantics

The "status" attribute is readOnly and computed by the server. Clients cannot set this value directly.

Status computation rules (evaluated in order):

1. If the resource has been soft-deleted (see Section 4.9): status = "revoked"

2. If the referenced subject.active is false: status = "suspended"

3. If current time < validity.validFrom: status = "pending"

4. If current time > validity.validTo: status = "expired"

5. Otherwise: status = "active"

Special cases:

* If validity.validFrom is null, the assignment is immediately eligible.

* If validity.validTo is null, the assignment does not expire.

Required status values:

* "active": assignment is currently effective

* "expired": validity window has ended

* "pending": assignment created but not yet effective

* "suspended": subject inactive or assignment temporarily disabled

* "revoked": assignment was explicitly withdrawn via DELETE operation

4.10. JSON Schema representation

```
{
  "id": "urn:ietf:params:scim:schemas:core:2.0:RoleAssignment",
  "name": "RoleAssignment",
  "description": "SCIM RoleAssignment Resource",
  "attributes": [
    {
      "name": "subject",
      "type": "complex",
      "description": "The subject receiving the role",
      "required": true,
      "mutability": "immutable",
      "returned": "always",
      "uniqueness": "none",
      "subAttributes": [
        {
          "name": "value",
          "type": "string",
          "description": "Identifier of the subject. If the subject is a resource, val
ue MUST equal the id attribute of that resource.",
          "required": true,
          "mutability": "immutable",
          "returned": "always",
          "caseExact": false,
          "uniqueness": "none"
        },
        {
          "name": "$ref",
          "type": "reference",
          "description": "URI to the subject resource, if the subject is a resource",
          "required": false,
          "mutability": "immutable",
          "returned": "default",
          "referenceTypes": ["User", "Group"],
          "uniqueness": "none"
        },
        {
          "name": "type",
          "type": "string",
          "description": "Subject type. If the subject is a resource, type MUST equal
the resource type name of that resource.",
          "required": false,
```

```
              "mutability": "immutable",
              "returned": "default",
              "caseExact": false,
              "canonicalValues": ["User", "Group"],
              "uniqueness": "none"
            },
            {
              "name": "display",
              "type": "string",
              "description": "A human-readable name for the subject",
              "required": false,
              "mutability": "immutable",
              "returned": "default",
              "caseExact": false,
              "uniqueness": "none"
            }
          ]
        },
        {
          "name": "scope",
          "type": "complex",
          "description": "The scope where the role applies",
          "required": true,
          "mutability": "immutable",
          "returned": "always",
          "uniqueness": "none",
          "subAttributes": [
            {
              "name": "type",
              "type": "string",
              "description": "Scope type. Common values include ’project’, ’tenant’, ’orga
nization’, ’application’, ’environment’",
              "required": true,
              "mutability": "immutable",
              "returned": "always",
              "caseExact": false,
              "uniqueness": "none"
            },
            {
              "name": "value",
              "type": "string",
              "description": "Provider-meaningful identifier of the scope. If the scope is
 a resource, value MUST equal the id attribute of that resource.",
              "required": true,
              "mutability": "immutable",
              "returned": "always",
              "caseExact": false,
              "uniqueness": "none"
            },
            {
```

```
            "name": "$ref",
            "type": "reference",
            "description": "URI to the scope resource, if the scope is a resource",
            "required": false,
            "mutability": "immutable",
            "returned": "default",
            "referenceTypes": [],
            "uniqueness": "none"
          },
          {
            "name": "display",
            "type": "string",
            "description": "A human-readable name for the scope",
            "required": false,
            "mutability": "immutable",
            "returned": "default",
            "caseExact": false,
            "uniqueness": "none"
          }
        ]
      },
      {
        "name": "role",
        "type": "complex",
        "description": "The role granted within the scope",
        "required": true,
        "mutability": "immutable",
        "returned": "always",
        "uniqueness": "none",
        "subAttributes": [
          {
            "name": "value",
            "type": "string",
            "description": "Stable identifier for the role. If the role is a resource, v
alue MUST equal the id attribute of that resource.",
            "required": true,
            "mutability": "immutable",
            "returned": "always",
            "caseExact": false,
            "uniqueness": "none"
          },
          {
            "name": "display",
            "type": "string",
            "description": "Human-readable name of the role",
            "required": false,
            "mutability": "immutable",
            "returned": "default",
            "caseExact": false,
```

```
            "uniqueness": "none"
          },
          {
            "name": "$ref",
            "type": "reference",
            "description": "URI to a role catalog entry or role resource, if available",
            "required": false,
            "mutability": "immutable",
            "returned": "default",
            "referenceTypes": [],
            "uniqueness": "none"
          },
          {
            "name": "type",
            "type": "string",
            "description": "Role type. If the role is a resource, type MUST equal the re
source type name of that resource.",
            "required": false,
            "mutability": "immutable",
            "returned": "default",
            "caseExact": false,
            "uniqueness": "none"
          }
        ]
      },
      {
        "name": "priority",
        "type": "integer",
        "description": "Conflict resolution priority. Higher values take precedence. Def
ault is 0.",
        "required": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "grant",
        "type": "complex",
        "description": "Assignment metadata",
        "required": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none",
        "subAttributes": [
          {
            "name": "source",
            "type": "string",
            "description": "Originating system or process",
            "required": false,
            "mutability": "immutable",
```

```
            "returned": "default",
            "caseExact": false,
            "uniqueness": "none"
          },
          {
            "name": "reason",
            "type": "string",
            "description": "Human-readable justification for the assignment",
            "required": false,
            "mutability": "readWrite",
            "returned": "default",
            "caseExact": false,
            "uniqueness": "none"
          },
          {
            "name": "approver",
            "type": "complex",
            "description": "The approver of this assignment",
            "required": false,
            "mutability": "immutable",
            "returned": "default",
            "uniqueness": "none",
            "subAttributes": [
              {
                "name": "value",
                "type": "string",
                "description": "Identifier of the approver. If the approver is a resourc
e, value MUST equal the id attribute of that resource.",
                "required": true,
                "mutability": "readWrite",
                "returned": "default",
                "caseExact": false,
                "uniqueness": "none"
              },
              {
                "name": "$ref",
                "type": "reference",
                "description": "URI to the approver resource, if the approver is a resou
rce",
                "required": false,
                "mutability": "readWrite",
                "returned": "default",
                "referenceTypes": ["User"],
                "uniqueness": "none"
              },
              {
                "name": "type",
                "type": "string",
                "description": "Approver type. If the approver is a resource, type MUST
equal the resource type name of that resource.",
                "required": false,
```

```
                "mutability": "readWrite",
                "returned": "default",
                "caseExact": false,
                "canonicalValues": ["User"],
                "uniqueness": "none"
              },
              {
                "name": "display",
                "type": "string",
                "description": "A human-readable name for the approver",
                "required": false,
                "mutability": "readWrite",
                "returned": "default",
                "caseExact": false,
                "uniqueness": "none"
              }
            ]
          }
        ]
      },
      {
        "name": "validity",
        "type": "complex",
        "description": "Validity window",
        "required": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none",
        "subAttributes": [
          {
            "name": "validFrom",
            "type": "dateTime",
            "description": "Start of validity window (RFC3339 format with optional timez
one offset)",
            "required": false,
            "mutability": "readWrite",
            "returned": "default",
            "uniqueness": "none"
          },
          {
            "name": "validTo",
            "type": "dateTime",
            "description": "End of validity window (RFC3339 format with optional timezon
e offset)",
            "required": false,
            "mutability": "readWrite",
            "returned": "default",
            "uniqueness": "none"
          }
        ]
```

```
      },
      {
        "name": "status",
        "type": "string",
        "description": "Computed lifecycle status",
        "required": false,
        "mutability": "readOnly",
        "returned": "default",
        "caseExact": true,
        "canonicalValues": ["active", "expired", "pending", "suspended", "revoked"],
        "uniqueness": "none"
      }
    ]
  }
```

4.11. Soft Delete and Audit Trail

RoleAssignment resources support soft deletion to maintain audit trails and compliance records.

When a DELETE operation is performed on a RoleAssignment: - The resource is NOT physically removed from the system - The status is set to "revoked" - The meta.lastModified timestamp is updated - The resource remains queryable but is typically excluded from standard operational queries

Servers SHOULD support filtering to exclude revoked assignments from normal operations: GET /RoleAssignments?filter=status ne "revoked"

Servers MAY implement retention policies to permanently delete revoked assignments after a configurable period (e.g., 90 days, 1 year) for compliance purposes.

Clients performing access control decisions MUST exclude assignments with status="revoked".

4.12. Complete Example

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:RoleAssignment"],
  "id": "assignment-12345",
  "externalId": "ext-assign-001",
  "subject": {
    "value": "alice@company.com",
    "$ref": "https://example.com/scim/v2/Users/alice",
    "type": "User"
  },
  "scope": {
    "type": "project",
    "value": "web-app-proj",
    "$ref": null
  },
  "role": {
    "display": "Developer",
    "value": "developer",
    "$ref": null
  },
  "priority": 100,
  "grant": {
    "source": "HR-System",
    "reason": "New team member onboarding",
    "approver": {
    "value": "manager@company.com",
    "$ref": "https://example.com/scim/v2/Users/manager",
    "type": "User",
    "display": "Alice Manager"
  }
  },
  "validity": {
    "validFrom": "2025-09-01T00:00:00Z",
    "validTo": "2026-09-01T00:00:00Z"
  },
  "status": "active"
}
```


Note: When this assignment is deleted via DELETE, the resource will remain with status changed to "revoked" and meta.lastModified updated, rather than being removed from the system.

5. Operations

RoleAssignment resources support standard SCIM operations with the following requirements:

5.1. Create (POST)

Servers MUST validate that subject, scope, and role reference valid resources or identifiers. Missing required attributes cause 400 Bad Request per RFC 7644.

5.2. Read (GET)

Servers MUST support retrieval of individual RoleAssignment resources by id.

5.3. Replace (PUT)

Servers MUST support full resource replacement. Attempts to modify immutable attributes (subject, scope, role, grant.source, grant.approver) MUST result in 400 Bad Request with appropriate error detail.

5.4. Patch (PATCH)

If PATCH is supported (per ServiceProviderConfig), servers MUST: - Prevent modification of immutable attributes - Validate that changes do not create invalid states - Return 400 Bad Request for operations attempting to modify immutable attributes

5.5. Delete (DELETE)

Servers MUST implement soft deletion: the DELETE operation sets status="revoked" and updates meta.lastModified rather than physically removing the resource. The resource remains queryable for audit purposes. Servers MUST immediately revoke the associated permissions in the target system.

5.6. Filter (GET with filter parameter)

If filtering is supported (per ServiceProviderConfig), servers MUST support filters on: - subject.value - scope.value - scope.type - role.value - status

Example: GET /RoleAssignments?filter=subject.value eq "alice@company.com" and status ne "revoked"

5.7. Pagination and Sorting

Servers MUST support pagination using startIndex and count parameters per RFC 7644.

If sorting is supported (per ServiceProviderConfig), servers SHOULD support sorting by: - meta.created - meta.lastModified - validity.validFrom - validity.validTo

5.8. Bulk Operations

Bulk operations MAY be supported per RFC 7644. Support is indicated in ServiceProviderConfig. Clients using bulk operations SHOULD populate externalId to enable idempotent retry of failed bulk requests.

5.9. Common Query Patterns

5.9.1. User’s assignments (excluding revoked):

GET /RoleAssignments?filter=subject.value eq "alice@company.com" and status ne "revoked"

5.9.2. All assignments in a scope:

GET /RoleAssignments?filter=scope.value eq "project-x"

5.9.3. Role discovery within a scope type:

GET /RoleAssignments?attributes=role,scope&filter=scope.type eq "project"

5.9.4. Expiring assignments:

GET /RoleAssignments?filter=validity.validTo le "2025-12-31T23:59:59Z" and status ne "revoked"

5.9.5. Audit trail - recently revoked assignments:

GET /RoleAssignments?filter=status eq "revoked" and meta.lastModified ge "2025-09-01T00:00:00Z"

5.9.6. All active assignments (operational query):

GET /RoleAssignments?filter=status eq "active" # Error Handling

RoleAssignment operations follow standard SCIM error handling per RFC 7644 Section 3.12. This section describes RoleAssignment-specific error conditions.

5.10. Invalid Validity Window

Status: 400 Bad Request scimType: invalidValue

Occurs when: - validity.validFrom is after validity.validTo - Date formats are invalid

Example: json { "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"], "status": "400", "scimType": "invalidValue", "detail": "validity.validFrom must be before validity.validTo" }

5.11. Invalid Resource Reference

Status: 400 Bad Request scimType: invalidValue

Occurs when: - subject.value references a non-existent or invalid resource - scope.value references a non-existent or invalid scope - role.value references a non-existent or invalid role

Example: json { "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"], "status": "400", "scimType": "invalidValue", "detail": "subject.value 'alice@example.com' does not reference a valid User resource" }

5.12. Duplicate Active Assignment

Status: 409 Conflict scimType: uniqueness

Occurs when attempting to create an assignment that would result in multiple active assignments with the same subject, scope, and role combination (unless distinguished by priority or validity windows).

Example: json { "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"], "status": "409", "scimType": "uniqueness", "detail": "Active assignment already exists for subject 'alice@company.com' with role 'Developer' in scope 'project-x'" }

5.13. Immutable Attribute Modification

Status: 400 Bad Request scimType: mutability

Occurs when attempting to modify immutable attributes (subject, scope, role, grant.source, grant.approver) via PUT or PATCH operations.

Example: json { "schemas": ["urn:ietf:params:scim:api:messages:2.0:Error"], "status": "400", "scimType": "mutability", "detail": "Attribute 'subject' is immutable and cannot be modified. Delete and recreate the assignment to change the subject." }

6. Backward Compatibility

Providers MAY continue to expose global roles via User.roles and coarse-grained membership via Groups. Providers SHOULD offer mapping guidance between legacy models and RoleAssignments.

During migration, providers are RECOMMENDED to: - Maintain both User.roles and RoleAssignments representations (dual-writing) - Use RoleAssignments for new scoped role grants - Gradually migrate existing Group-based role patterns to RoleAssignments - Preserve audit history when converting legacy role assignments

Clients transitioning to RoleAssignments SHOULD: - Query both User.roles and RoleAssignments during migration periods - Implement fallback logic for providers not yet supporting RoleAssignments - Use externalId to correlate legacy role grants with new RoleAssignment resources

7. Security Considerations

* RoleAssignment creation and modification are privileged operations.

* Servers MUST validate references to prevent privilege escalation.

* Lifecycle events SHOULD be logged with actor and justification.

* Replay and bulk abuse MUST be mitigated with rate limiting and idempotency.

* Soft deletion preserves audit trails but increases storage requirements and query complexity. Servers MUST ensure that revoked assignments are excluded from authorization decisions, even though they remain in the data store. Servers SHOULD implement periodic cleanup of old revoked assignments according to organizational retention policies. Access to revoked assignments SHOULD be restricted to auditors and compliance personnel.

8. Privacy Considerations

RoleAssignments may expose organizational structures and access patterns. Sensitive metadata SHOULD follow least-privilege disclosure.

9. IANA Considerations

This specification requests registration of the following SCIM schema:

* URN: urn:ietf:params:scim:schemas:core:2.0:RoleAssignment
* Specification: this document
* Contact: IETF SCIM Working Group
* Change Controller: IESG

Experimental namespace MAY also be used:

* URN: urn:ietf:params:scim:schemas:extension:role:1.0:RoleAssignment

URNs are assigned and interpreted in accordance with [RFC8141].

10. Conformance

10.1. Server

A conformant SCIM server implementation:

* MUST implement RoleAssignment resource type and advertise it in /ResourceTypes
* MUST validate subject, scope, and role references on all operations
* MUST enforce authorization on RoleAssignment operations
* MUST support GET, POST, PUT, DELETE operations
* MUST enforce immutability of subject, scope, role, grant.source, and grant.approver attributes
* MUST implement soft deletion: DELETE operations set status="revoked" rather than removing resources

* MUST exclude status="revoked" assignments from authorization decisions

* MUST support filtering on subject.value, scope.value, scope.type, role.value, and status (if filtering is supported per ServiceProviderConfig)

* SHOULD support PATCH operations per RFC 7644

* SHOULD support sorting by meta.created, meta.lastModified, and validity fields (if sorting is supported per ServiceProviderConfig)

* SHOULD support bulk operations per RFC 7644

* MAY implement retention policies for permanent deletion of old revoked assignments

10.2. Client

A conformant SCIM client implementation:

* MUST construct RoleAssignments with all required attributes per the schema

* MUST process error responses per RFC 7644 Section 3.12

* MUST NOT use assignments with status="revoked" for access control decisions

* MUST NOT attempt to modify immutable attributes (subject, scope, role, grant.source, grant.approver)

* SHOULD use externalId for idempotency, especially in bulk operations

* SHOULD honor ETag preconditions for conditional updates

* SHOULD filter out revoked assignments in operational queries using filter=status ne "revoked"

* MAY query revoked assignments for audit and compliance purposes

11. Change Log

11.1. draft-poreddy-scim-role-assignment-01

Second version. Major changes based on working group feedback: - Added complete JSON Schema definitions with mutability specifications - Implemented soft deletion for audit trail preservation - Made subject, scope, and role immutable; priority and validity mutable - Restructured grant.approver as complex attribute with full reference pattern - Added explicit resource mutability section and design rationale - Updated all attribute tables to reflect immutability decisions - Enhanced error handling with RoleAssignment-specific scenarios - Clarified Operations section with ServiceProviderConfig references - Added comprehensive conformance requirements for servers and clients

11.2. draft-poreddy-scim-role-assignment-00

Initial version. Defines RoleAssignment schema, attributes, and operations. Adds priority, complete example, error examples, and query patterns. Includes error handling, backward compatibility, and IANA registration. Updates BCP14 boilerplate; replaces non-ASCII; converts ASCII tables to RFCXML tables; updates Roles/Entitlements reference to -02; cites RFC8141.

12. Author’s Address

Prithvi Poreddy Email: prithvikrishnab4u@gmail.com

13. References

13.1. Normative References

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, DOI 10.17487/RFC7643, September 2015, <https://www.rfc-editor.org/info/rfc7643>.

[RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <https://www.rfc-editor.org/info/rfc7644>.

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <https://www.rfc-editor.org/info/rfc8141>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[GITLAB-SCIM]
GitLab Documentation, "Set up SCIM for GitLab groups", September 2025, <https://docs.gitlab.com/administration/settings/scim_setup/>.

[TANIUM-RBAC]
Tanium Documentation, "Role-based access control in the Tanium Console", September 2025, <https://help.tanium.com/bundle/ug_console_cloud/page/platform_user/console_roles.html>.

[AZURE-SCIM]
Microsoft Learn, "Issue provisioning multiple roles to a SCIM app", September 2025, <https://learn.microsoft.com/en-us/answers/questions/1632657/issue-provisioning-multiple-roles-to-a-scim-app>.

[SCIM-ROLES-ENTITLEMENTS]
Zllner, C., "Roles and Entitlements Extension for SCIM", Work in Progress, Internet-Draft, draft-zollner-scim-roles-entitlements-extension-02, June 2025, <https://datatracker.ietf.org/doc/html/draft-zollner-scim-roles-entitlements-extension-02>.

Author’s Address

Prithvi Poreddy
Individual Contributor
United States of America
Email: prithvikrishnab4u@gmail.com