

SCIM Interoperability Profile

draft-zollner-scim-interop-profile

IETF 125 - March 2026

The Problem

- Many features in SCIM are optional to implement
- SCIM provides too many ways to accomplish some actions
- “B2B SaaS” SCIM service provider implementers struggle with implementing the right features to be compatible with the IdPs (SCIM clients) that their customers use
- Each IdP’s SCIM client implementation is different and creates unique requirements for interoperability with service providers

Proposed Solution In This Draft

- Draft sets modern requirements aiming to standardize what SCIM integration between IdPs and B2B SaaS applications looks like
- Draft includes requirements that:
 - Define a subset of SCIM PatchOp syntaxes that SCIM clients must use
 - Define a minimum set of required attributes for users and groups
 - Defines minimum requirements for filter and pagination support
 - Explicitly call out certain requirements that are already present in RFC 7643 & RFC 7644 but are commonly implemented incorrectly
 - Prohibit certain features and/or behaviors in SCIM 2.0

HTTP PATCH Simplification

6.4.2.1. Singular Attributes (Simple and Complex)

Simple Attribute Operation Equivalence

For singular simple attributes (e.g., `userName`, `active`), Service Providers **MUST** treat add and replace as functionally equivalent.

Complex Sub-attribute Targeting

When only intending to modify the value of a specific sub-attribute of a complex attribute, Clients **SHOULD** target that sub-attribute using dot-notation (e.g., path: `"name.givenName"`).

Complex Attribute Operations

The following rules apply to each PATCH operation type:

Replace

If a Client targets a singular complex attribute in its entirety (e.g., path: `"name"`) using the replace operation, the value **MUST** be a JSON object containing all sub-attributes the Client intends to persist.

Add

If a Client targets a singular complex attribute in its entirety using the add operation, the value **MUST** be a JSON object. The Service Provider **MUST** perform a partial merge.

HTTP PATCH Simplification

6.4.2.2. Multi-valued Attributes (Simple and Complex)

Collection-Level Operations

The following rules apply to each PATCH operation type:

Replace

If a Client targets a multi-valued attribute path without a filter using the replace operation, the value MUST be an array.

Add

If a Client targets a multi-valued attribute path without a filter using the add operation, the value MUST be an array.

Filtering Constraints

When using a filter to target an element within a multi-valued complex attribute:

Sub-attribute Requirement

The path MUST target a specific sub-attribute. (e.g.: “path” : “emails[type eq \”work\”].value”)

Prohibition

Targeting the element object itself is PROHIBITED. (e.g.: “path” : “emails[type eq \”work\”]”)

Deprecated / Prohibited Features and Behaviors

- User “password” attribute
- HTTP PUT
- Silently ignoring unknown attributes in POST/PATCH requests and returning a successful response
- Conditional versioning-related HTTP headers (if-match, if-modified-since..)
- Filters in the request URL for HTTP methods other than GET
 - E.g.: PATCH /Users?filter=.. to apply the update in the request body to all matching resources

Next Steps

- Gather feedback
- Continue iterating on requirements

Feedback?