What is SCIM?

“System for Cross-domain Identity Management”

• Protocol designed for sharing data across identity contexts
• Consists of the communication protocol and the core schema
  • Allows for extensibility

“builds upon experience with existing schemas and deployments, placing specific emphasis on simplicity of development and integration” rfc7643

“In essence, make it fast, cheap, and easy to move users in to, out of, and around the cloud.” rfc7642
Why SCIM?

• Abstracts away underlying data structure
• Treats every site/domain/whatever the same, enables scale
last_name: Jane Smith
email: jsmith@example.org
home_phone: 432-123-4566

{ "schemas":
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User" ],
"id": "43234723-7778-4534-567857657654",
"userName": "jsmith@example.org",
"name": {
  "formatted": "Ms. Jane Smith",
  "familyName": "Smith",
  "givenName": "Jane"
 },
"phoneNumbers": [
  {
    "value": "432-123-4566",
    "type": "home"
  }
],
"emails": [
  {
    "value": "jsmith@example.org",
    "type": "work",
    "primary": true
  }
] }
The SCIM Protocol RFC 7644

• Simple RESTful APIs designed for developer ease of use
  • uses HTTP methods (verbs) GET, POST, PUT, PATCH, DELETE for creating and modifying the data via SCIM standard JSON payloads to the resource endpoints (nouns)
• Standardizes methods for clients to communicate with servers (Service Providers)
• Provides a client the ability to discover resources and server config via three discovery endpoints

Service Provider: An HTTP server (rfc2616) which exposes the standard SCIM APIs for Create, Read, Update & Delete operations for a SCIM Client
## Available Standard SCIM Endpoints

<table>
<thead>
<tr>
<th>Discovery</th>
<th>Endpoint</th>
<th>Description</th>
<th>Supported REST Verbs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/Users</td>
<td>Create, Read, Update, Delete on user data</td>
<td>GET, POST, PUT, PATCH, DELETE</td>
</tr>
<tr>
<td></td>
<td>/Groups</td>
<td>Create, Read, Update, Delete on group data (including membership)</td>
<td>GET, POST, PUT, PATCH, DELETE</td>
</tr>
<tr>
<td></td>
<td>/Me</td>
<td>Create, Read, Update, Delete on users own data</td>
<td>GET, POST, PUT, PATCH, DELETE</td>
</tr>
<tr>
<td>✔</td>
<td>/Schemas</td>
<td>Retrieve available schemas</td>
<td>GET</td>
</tr>
<tr>
<td>✔</td>
<td>/ResourceTypes</td>
<td>Retrieve available resource types</td>
<td>GET</td>
</tr>
<tr>
<td>✔</td>
<td>/ServiceProviderConfig</td>
<td>Retrieve available config on the server (ServiceProvider)</td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td>[prefix]/.search</td>
<td>Read and returns queried data</td>
<td>POST</td>
</tr>
<tr>
<td></td>
<td>/Bulk</td>
<td>Contains series of operations to the server in bulk</td>
<td>POST</td>
</tr>
</tbody>
</table>
## Examples of REST operations

<table>
<thead>
<tr>
<th>REST Operation</th>
<th>Resource</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/Users</td>
<td>Creates new resource</td>
</tr>
<tr>
<td>GET</td>
<td>/Users</td>
<td>Returns all resources</td>
</tr>
<tr>
<td>GET</td>
<td>/Users?filter=userName eq “<a href="mailto:user@domain.com">user@domain.com</a>”</td>
<td>Returns all resources matching filter</td>
</tr>
<tr>
<td>GET</td>
<td>/Users/123</td>
<td>Returns resource with matching id value</td>
</tr>
<tr>
<td>PUT</td>
<td>/Users/123</td>
<td>Updates all attributes on resource</td>
</tr>
<tr>
<td>PATCH</td>
<td>/Users/123</td>
<td>Updates only specified attributes on resource</td>
</tr>
<tr>
<td>PATCH</td>
<td>/Users?filter=userType eq “Intern”</td>
<td>Updates all resources matching filter</td>
</tr>
<tr>
<td>DELETE</td>
<td>/Users/123</td>
<td>Deletes resource with matching id value</td>
</tr>
</tbody>
</table>
The SCIM Core Schema RFC 7643

• Defines a minimal common set of attributes representing user and group data along with an enterprise extension for user data

• Provides a method for organizations to extend SCIM
  • Schemas
  • Resource types

• Custom schemas may be permanently registered with IANA
  https://www.iana.org/assignments/scim/scim.xml
Example SCIM User: GET /Users/123

**name**: single-valued complex attribute
The components of the user's name

**phoneNumbers**: multi-valued complex attribute with type for meaningful value to human user

**emails**: multi-valued complex attribute
Canonicalized value with type to provide meaningful value to human user

**Resource Schema Representation:**
https://datatracker.ietf.org/doc/html/rfc7643#section-8.7.1

```json
{
    "id": "123",
    "name": { "formatted": "Ms. Jane Smith", "familyName": "Smith", "givenName": "Jane" },
    "phoneNumbers": [ { "value": "432-123-4566", "type": "home" } ],
    "emails": [ { "value": "jsmith@example.org", "type": "work", "primary": true } ]
}
```
Attribute Data Types

• String
• Boolean
• Decimal
• Integer
• DateTime
• Binary
• Reference (a URI to a resource)
• Complex (a composition of simple attributes)
And we all lived happily ever after, until...

• **Usability**
  • Spec ambiguity led to multiple ways to implement causing interoperability challenges and data corruption
  • Limited guidance on groups, roles, entitlements
  • Common attributes not fully defined in 7643

• **Improvements**
  • Bulk operations are not asynchronous
  • Pagination limitations
  • Core Schema Extensibility

• **New and Emerging**
  • Limited privileged access management (in draft)

Other... *Papercuts*
Goals of SCIM NextGen

• Improve the overall best practices and guidance
  • Profiling SCIM relationship with other Identity Protocols
• Account State and Soft Deletion
• Reduce areas of ambiguity and provide more prescriptive examples
• Schema enhancements for HR, Enterprise Group, privileged access management
• Advanced automation scenarios
• Enhanced data handling for larger data sets

https://datatracker.ietf.org/doc/charter-ietf-scim/