Change Detection

Anjali Sehgal (AWS) and Danny Zollner (Microsoft)

Goal

Provide a scalable and highly accurate method of change detection that allows a SCIM client to retrieve the current state of all resources that have changed since a prior point in order to perform a recurring incremental retrieval of data.

Some of the ***possible use cases*** where this can be used:

1. **Reconciliation System:**
    a. There is a need to build a reconciliation system to identify undetected diverging data between a SCIM Client and SCIM Server in order to prevent undesirable authorization decisions based on incorrect data.
    b. The data divergence detection may be used for reporting purposes or may be extended to either trigger provisioning of those resources in the target system or pulling changes from the target system into the source.
    c. Without a set of end-to-end reconciliation processes, service providers cannot confidently say that there are no divergence-caused security incidents.

2. **Incremental Synchronization** where client pulls data from the server.
    a. Example: Identity Provider pulling data from an HR system.

Requirements

- Resources modified since a specific point can be returned by query
- Current state of resources returned
- Able to convey that a previously existing resource was deleted since specified point
- Able to convey changes to group memberships
- Performant at large scale with accurate results
- … others?

Approach 1: Change Detection Using Timestamp-based Filter

Flow of Sync Events

1. Client starts with a full retrieval of relevant data.

   GET /Users

2. Client queries the server again, along with filter on meta.lastModified to request all resources created, updated since the last query.

   GET /Users?count=100&filter=(meta.lastModified gt ${now} - {syncInterval}) AND (meta.lastModified lt ${now})

3. To include deleted records in the response use new query parameter “includeDeleted”

   GET /Users?includeDeleted&count=100&filter=(meta.lastModified gt ${now} - {syncInterval}) AND (meta.lastModified lt

   ${now})

Approach 1: Change Detection Using Timestamp-based Filter

Benefit: Simple to implement using underlying search framework for many systems

Limitation: Using timestamp has underlying risks related to time drift

- T**ime drift on the SCIM server** - a query starting at 12:00 may miss a change that happened at 12:00:00002 but due to time drift between backend server components landed at 11:59:99998
- **Time drift between client and server:** How to validate if 12:00:000 on the client is 12:00:000 on the server

Mitigations

1. Client/Server keep track of client/server time difference.
2. Clients can ddd an offset/overlap of search periods. This introduces possibility of resources being returned in multiple successive queries without having been changed.

GET Users?includeDeleted&count=100&filter=(meta.lastModified gt ${now} - {syncInterval}) AND (meta.lastModified lt ${now}-{overlap})

Approach 2: Change Detection using Watermark-based Approach

Usage of an opaque artifact generated by a SCIM server which provides a point of reference value that can be used to identify resources created, updated or deleted after the point represented by the value.

Flow of Sync Events

1. Client starts with a full retrieval of relevant data

   *GET /Users?**deltaQuery**=**true**&count=50*

   The server as part of the response to this request returns an artifact called **nextDeltaToken.**

2. The value is provided by the client on a subsequent GET request to only retrieve resources created, updated or deleted after the point represented by the value.

   GET Users?deltaQuery=true&deltaToken="VTHKLOUTREO"&count=50

3. This process can be repeated continuously to allow the client to efficiently keep track of changes

Approach 2: Change Detection using Watermark-based Approach

Benefits

- More accurate and efficient by avoiding timestamp-based issues.
- Reduces time variables that the client is required to track and adjust for.
- Provides flexibility to the server implementers to map the delta token internally to a timestamp, a location in a change log etc.

Limitations

- It is more complex solution to implement by SCIM Servers.
- Client's loss of recent watermark values may require repeating full retrieval of data
- Does not provide flexibility to get changed records within a specific period of time.
    - This may be required in cases where client may want to get changes within a specific period of time after a bug is fixed ensure there is no data divergence between that period of time.

Open Items we are seeking input on

- Optimal method of specifying the “point” at which the next incremental sync starts at
  - Approach 1: Datetime?
  - Approach 2: Watermark?
- Should both be supported or only one should be supported ?

- How to communicate changes to group memberships
  - Limit to “this group has changed” and force a GET of the group?
  - New protocol for responses - possibly modeled on PATCH add/remove/replace syntax?
  - Solve in combination with draft addressing group membership pagination?
- What is required for SCIM’s new or existing protocol to allow for 100% accurate results?
- Is there value in supporting “GET <resource> where <attribute> has changed”?