Cursor-based Pagination

https://datatracker.ietf.org/doc/draft-ietf-scim-cursor-pagination/

Administrative

- Matt Peterson (original author) stepping back from draft
- Danny Zollner, Dean H. Saxe, Anjali Sehgal and David Brossard taking over future authoring/editing responsibilities

Feedback from Call for Adoption

- Concerns:
  - Denial of Service via server resource exhaustion
  - Security concerns if pagination is implemented in a way that is stateful and locks resource records while the cursor is open

- Response:
  - Explicit guidance that cursor-based pagination is not expected to be stateful
    - Lack of requirement for stateful snapshot of results reduces opportunity for DoS/resource locking issues
  - Additional mention in security considerations for both of the above as well

Feedback from Call for Adoption - Continued

- Concerns:
  - Additional requirements of change detection/delta query capabilities

- Response:
  - Change detection/delta query not a requirement, but a valuable complementary functionality for large-scale client/server “pull” activity

Proposed Text Addition to Draft under Introduction Section 1

"*Similar to index-based pagination, cursor-based pagination is not stateful, clients MUST be prepared to handle changes in results when the underlying data set is changing. New objects can be added, existing objects can be deleted or updated while the client is paginating through the dataset. Cursor based pagination unlike index-based pagination, eliminates the possibility of skipped objects or duplicated objects while paginating frequently changing large sets of data.*

For example, let's say the client is paginating through 5M users with a page size of 100. Let's say that while the client is processing the content of page 2, a user object that occurred on page 1 is deleted. The user object at offset 301 will now shift to position 300. As a result, with index-based pagination, the client will not be able to see this object when it retrieves page 3, unless it retrieves page 2 again. In contrast to this with cursor-based pagination the client will be able to fetch this object as the next cursor on page 2 will point to record stored at offset 301 before the delete was performed.

Similarly, if a new user object is inserted at offset 210, the position of object retrieved as part of page 2 and stored at offset 300 will change to 301. With index-based pagination clients will retrieve this object again in page 3 because its offset changed to 301, however, with cursor-based pagination this object will not reappear in page 3 as the next cursor of page 2 points to object that was stored at offset 301 before the insert of the new record was committed."

Upcoming Revisions

- Soon™: -01, adds mentioned statefulness/security guidance
- Later: -02, address feedback from Httpdir review of draft