Handling Web Cache Validation with ETags

ETags *may* be used to reduce bandwidth usage by preventing the contents of an unmodified resource from being returned. An Ed-Fi REST API *should* support such cache validation through the use of the “If-None-Match” request header. If the ETag value supplied in the request header still matches the existing resource, the API *may* respond with a 304 (Not Modified) status code with no response body, rather than a 200 (OK) with the resource content.

Implementing Bulk Operations

A REST API is one mechanism for bulk data loading into a data store. While other possibilities abound, when a REST API bulk load API is provided, the considerations in the following sections apply.

Security

The bulk data portion of the Ed-Fi REST API *may* use system user authentication, but *must* use application authentication. In addition, installations *should* use one or more additional authentication mechanisms (such as client IP address restrictions, pre-shared certificates, or VPN access) to further secure the bulk data transfer APIs where possible.

Atomicity of Data

REST APIs are based on simple HTTP verbs and use the HTTP protocol for data transfer. The HTTP PUT and POST verbs packet sizes can be limited by the web server and client configurations. There is no size limit in the HTTP protocol itself for PUT or POST operations, however practical limitations exist.[16]

Packet sizes *should* be limited to a reasonable maximum based on the capabilities of the host system. All resource transactions are considered upserts[17] and are idempotent; a single resource failure within a packet does not invalidate the packet, but subsequent Resources from that packet are processed. Failed uploads *may* be resubmitted, but are treated like a new submission.

Data Ordering

The potentially large quantity of transactions contained in a batch operation introduces a design trade-off between reduced system responsiveness while the batch operations are run and transactional operations are halted, or transactional operations becoming intermingled with batch transactions.

The selection of a batch load strategy is an implementation concern. Either approach has merit under appropriate circumstances.

Scalability

Any API has the potential to be overwhelmed with operation requests where the API depends upon data- or calculation-intensive resources. Relational data stores, and most especially the referential integrity checks generally employed in their designs, are data-intensive operations. Likewise, the other ACID (Atomicity, Consistency, Isolation, Durability) properties of a relational data store transaction impose computation and data requirements that potentially monopolize the repository resources, leading to intrinsic scalability concerns.

While NoSQL data stores do not have the same intrinsic design constraints, the relational nature of the resources in the Ed-Fi data model means that incoming data *must* be verified for referential integrity before the data is accepted as correct and is stored in a manner that is subsequently accessible to API clients.

Whether relational integrity is verified using relational schema or programmatically during bulk data ingestion, this process can be resource-intensive in production systems. Bulk data packages *should* be accepted for processing without a validity check. Asynchronous processing *may* then perform schema validation on the supplied data, and *must* perform basic data type validation and internal referential integrity checks before accepting the data and subsequently making it accessible to API clients.

The batch *should* be processed sequentially. Any operation failures within the batch *should not* halt processing of valid operations. The submitter *should* be notified of any operation failures asynchronously.

The exact mechanisms for queuing are left to the implementation, but *should* ensure transactional integrity, first-in/first-out (FIFO) ordering, and minimize long connections and resource locks.

---

[16] Internet Information Server and Apache have a default file upload limit of 2GB. Many browsers and frameworks often have this value as a configurable limit as well.

API Guidelines Contents

Find out more about the Ed-Fi API Design & Implementation Guidelines:

[17] A term representing a combination of an UPDATE (to existing records) and INSERT (for new records) functionality, detailed here.