



# API Design

When designing REST APIs, there are some nuanced design decisions. The following decisions work well in our case, which notably consider stratus' requirements on user-data and stratus' purpose as an evaluation tool:

- User-data API calls (those you may GET/PUT/DELETE) must not contain computed data, as then the `versionId` no longer determines the payload (i.e. there could be different payloads with equal `versionId`). This is surprising behavior to the developers, mixes up-to-date and out-of-date data and breaks caching. Instead, always use separate evaluation calls for computed data.
- Evaluation API calls (contain computed data) should not contain meta-data of the referenced entities, as this bloats the response size, while it requires all APIs to change when features are introduced like “filter by object groups everywhere”. Instead, only the `id` of the referenced entities should be contained, and then the consumer has to combine the data retrieved from the user-data API calls.

Older APIs may not conform to these design principles. The following issues are open in that regard:

-  [STRA-2218: \(FE\) Clean up object evaluation API behaviors](#) OFFEN
-  [STRA-2593: \(FE/BE\) Retire meta-data in evaluation API responses](#) OFFEN

Note that the frontend state architecture assumes this API design.