HHS 2020
Person-Centered
Health and Human Services

Preventing Service Sprawl

Vladislav Vilensky
Enterprise Architect
03/07/2019

Are we ready for 1K of business services?

Pick any business noun, let’s say Claim:

- Some clients want very little info – just claim status
- Some clients want the whole set of 100’s of attributes and those of related objects – claim details with details about all associated providers
- Yet others want some arbitrary mix of Claim attributes for some obscure purpose
- Some cant move forward until service has responded – synchronous clients
- Other clients want to be notified when requested info is available without waiting for response – asynchronous clients
- And yet other clients don’t want to process messages with claim details, they want files with information delivered to them

The number of Entity Type, data granularity and access mechanism combinations and permutations is staggering!

HHS 2020
Person-Centered
Health and Human Services

Can we limit ourselves to a set of SCRUD services per entity type?

(1 search + 1 get + 1 update + 1 delete service per Entity type)

X number of access mechanisms

X number of Entity types

Fairly high number of services, but better than the “unlimited sprawl” alternative

Why not do this?

- A Get <insert Entity type here> service would return the entire Entity to every caller. This wastes resources and diminishes performance.
- We can do better and hopefully without making architecture overly complex

HHS 2020 Person-Centered Health and Human Services

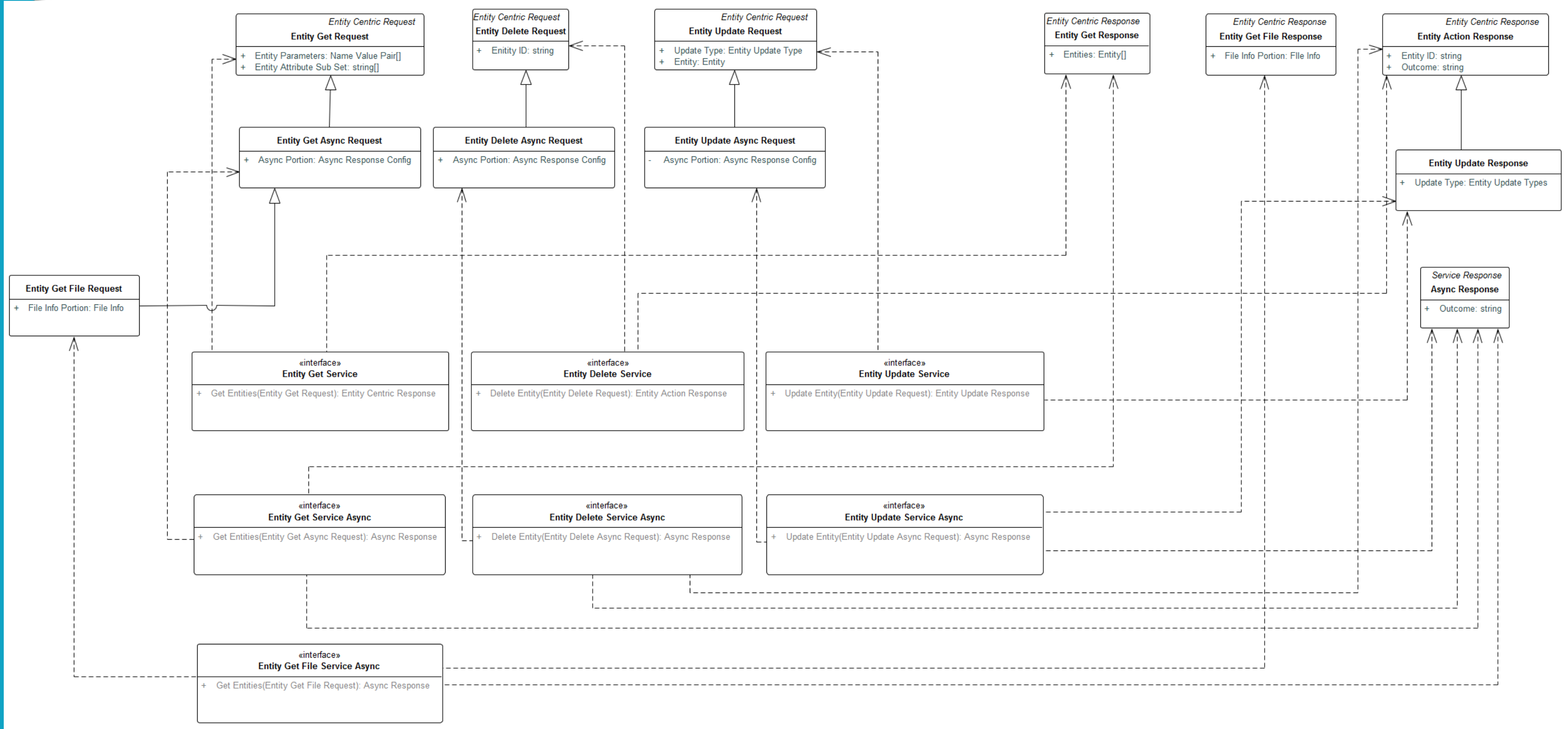
Can we limit ourselves to only 7 canonical services?

1. Synchronous Message-based Entity Get
2. Asynchronous Message-based Entity Get
3. Asynchronous File-based Entity Get
4. Synchronous Message-based Entity Update
5. Asynchronous Message-based Entity Update
6. Synchronous Message-based Entity Delete
7. Asynchronous Message-based Entity Delete

The 7 in words

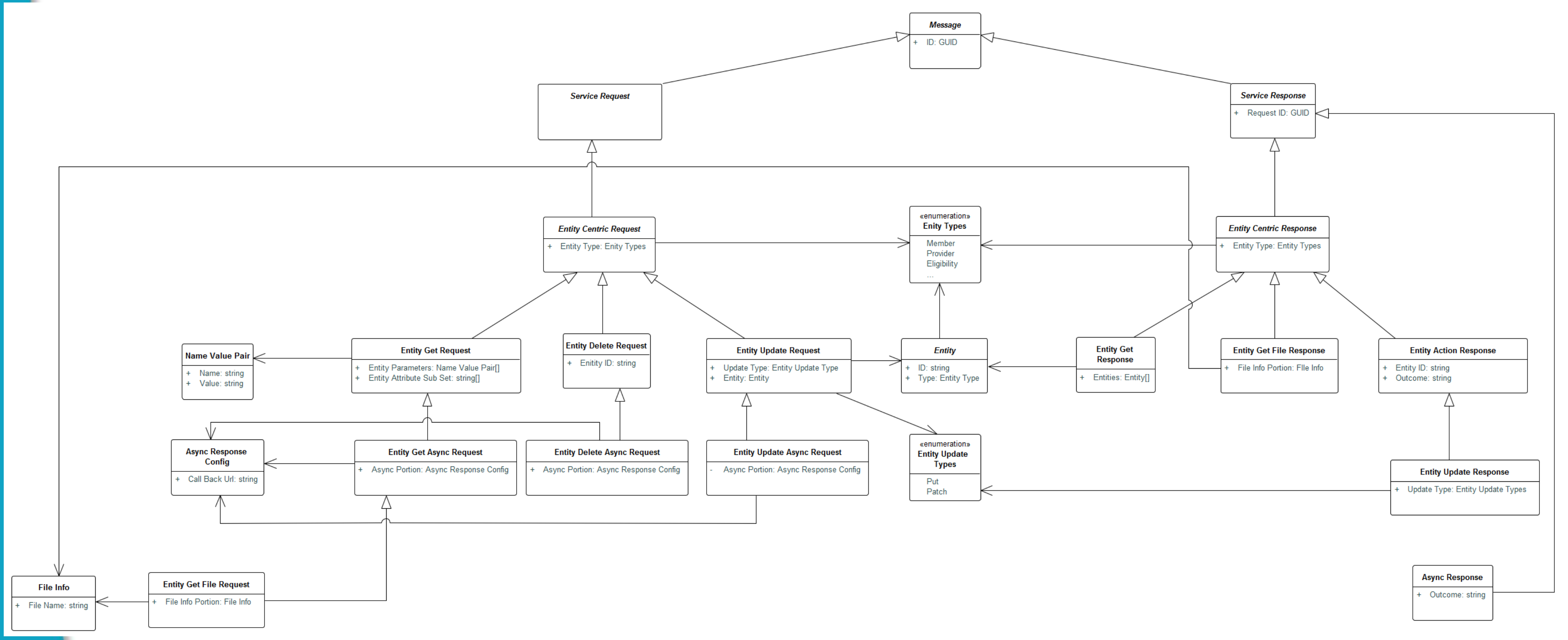
- Services will receive request information as a single string parameter with string contents encoded either as XML or as JSON
- Responses will be rendered as a single string encoded the same way as the corresponding request. In other words, XML request results in XML response, JSON request results in JSON response
- Service endpoints will be accessible through HTTP with a single request parameter (see previous bullet point)
- All requested information will always be contained in the request string. We will not be submitting requests with information contained in a file placed in some predetermined location.
- We will support synchronous and asynchronous invocation of services that exchange messages, and asynchronous invocation of services that produce files with service output to be picked up by the service consumer whenever ready

The 7 in UML


Entity Centric Request
Entity Get Request
+ Entity Parameters: Name Value Pair[]
+ Entity Attribute Sub Set: string[]
Entity Centric Request
Entity Delete Request
+ Entity ID: string
Entity Centric Request
Entity Update Request
+ Update Type: Entity Update Type
+ Entity: Entity
Entity Centric Response
Entity Get Response
+ Entities: Entity[]
Entity Centric Response
Entity Get File Response
+ File Info Portion: File Info
Entity Centric Response
Entity Action Response
+ Entity ID: string
+ Outcome: string
Entity Get Async Request
+ Async Portion: Async Response Config
Entity Delete Async Request
+ Async Portion: Async Response Config
Entity Update Async Request
- Async Portion: Async Response Config
Entity Update Response
+ Update Type: Entity Update Types
Entity Get File Request
+ File Info Portion: File Info
Service Response
Async Response
+ Outcome: string
«interface»
Entity Get Service
+ Get Entities(Entity Get Request): Entity Centric Response
«interface»
Entity Delete Service
+ Delete Entity(Entity Delete Request): Entity Action Response
«interface»
Entity Update Service
+ Update Entity(Entity Update Request): Entity Update Response
«interface»
Entity Get Service Async
+ Get Entities(Entity Get Async Request): Async Response
«interface»
Entity Delete Service Async
+ Delete Entity(Entity Delete Async Request): Async Response
«interface»
Entity Update Service Async
+ Update Entity(Entity Update Async Request): Async Response
«interface»
Entity Get File Service Async
+ Get Entities(Entity Get File Request): Async Response

HHS 2020
Person-Centered
Health and Human Services

Message Class hierarchy



HHS 2020
Person-Centered
Health and Human Services

The 7 in action

Synchronous entity get

- Satisfies 2 purposes: Entity Search and Entity Get
- The service will accept entity get request parameter consisting of selection criteria. Current design shows an array of name-value pairs. This model may be overly simplistic and would require an upgrade to more of a criteria object that can combine complex selection clauses joined by logical expressions AND, OR and matching logic like BEGINS WITH etc.
- The second portion of the request is an array of XPath expressions encoded as strings that specify schema elements of the requested entity that needs to be brought back
- The service would be invoked synchronously and the client would be blocked until an instance of Entity Get Response is returned

Asynchronous entity get

- The asynchronous version of entity get service will return immediately an instance of asynchronous response object that provides information to the caller about acceptance (or rejection) of the request. Asynchronous request is derived from the synchronous request with addition of call-back URL, which the service implementation is going to call once the results have been obtained and properly encoded.

Asynchronous file-based entity get

- The file-based version of the entity get service is invoked asynchronously, and rather than eventually calling the client with a message containing the requested entity instances, it generates a file containing requested instances and simply informs the caller about availability of the file

Synchronous entity delete

- We made a decision that we will not allow for bulk deletes of entities, and each delete request must be based on a unique identifier for the entity that is to be deleted. The caller is blocked until a delete response message is sent back.

Asynchronous entity delete

- Asynchronous version of delete releases the client immediately and calls back with delete completion on the specified call-back URL

Synchronous entity update

- Like deletes, updates are geared at one entity at a time. The request message must contain the entity that is being updated. A service implementation will examine which attributes are present in the supplied entity object, thus no need for additional XPath attributes explaining what information is present in the request. We need to investigate further whether we want to support the granularity of update of PATCH, equivalent to a database update statement, or PUT, equivalent of delete if exists plus insert database statements.

Asynchronous entity update

- Asynchronous entity update works in the same fashion as all other asynchronous versions

HHS 2020
Person-Centered
Health and Human Services

Much more work to do

- What complexities do we bring to the world of authorization by not having different service URLs for which permissions can be created, but having to analyze request contents to decide whether given user has the right to perform the operation on the requested entity type. Because we only have 1 function for each of the SCRUD operations, how do we find out if the user is entitled to search for claims vs. search for members. The obvious answer is it’s driven by business rules – if so, what kind of implementation would this lead to?
- Performance implications of additional context analysis. Do we have enough compute to support our projected usage volumes?
- Of the 7 services we specified, not all are equally important. An opinion has been expressed that file-based get is of the lowest priority because it’s a mechanism that’s discouraged in our architecture. What is the roadmap of priorities?
- Should we support file-based requests? For e.g. a mass claim update coming in from the FS vendor in a file may be fed to a service as a parameter. Under the current architecture, we probably would have a file ingestion service (not part of the entity-centric services) that would process contents of such a file and call the entity-centric service for each record. Is that acceptable?

Thank You!

HHS 2020
Person-Centered
Health and Human Services