Reducing overhead for remote service calls

The biggest overhead associated with calling annotators configured as services is due to CAS serialization and communication. In order to minimize this overhead, we are planning to implement two new features: "CAS projections" and "delta CAS".

CAS projection will allow only the subset of a CAS needed by an annotator service to be serialized and delivered to the service. For the reply from processing, delta CAS will enable returning only the changes to the CAS made by the annotator service.

Our design objective is to make these changes be transparent to annotator developers, but there are some cases where component descriptor changes will be needed to be able to best take advantage of these new capabilities.

Details on the proposal are in the following sections.

1. Service registration by the client is done by making a getMeta call, requesting metadata describing the service. The getMeta response will be modified to include service preferences for the input CAS format. A new component descriptor parameter will specify that the service accepts CAS projections. CAS projections as input will not work if the input capabilities do not include all the types & views required by the annotator. To avoid the possibility of breaking existing services, CAS projections will not be enabled by default.

~~Similarly, a new property will be added to the getMeta call indicating the desire for a delta CAS reply format.~~

A new property will be added to the processCAS call indicating the desire for a delta CAS reply format since the service will handle requests from multiple clients. Older clients not supporting delta CAS replies will of course not ask for delta CAS and will be able to utilize new services .Older services will ignore requests for delta CAS and continue to return a complete CAS. In the case where the client sends a "CAS projection" in the processCAS call, the new property will indicate that the client **requires** that the service return a delta CAS.

2. A CAS projection is the subset of a CAS defined by the cross product of the types and views (Sofas) declared in the service's input capabilities. The option to generate CAS projection will be built into XMI serialization code. Proposed rules for creating a CAS projection are:

- The _InitialView is always included in the list of views to be sent.
- All specified types (and their subtypes) which are indexed in the specified views are added to the projection. Note that specifying uima.cas.TOP will result in all indexed FS for the specified views being included.
- Non-indexed FS which are referenced by FS previously added to the projection are also added, if their type is specified, but with the following exception: any **annotation FS** belonging to a view not in the specified list of views is not added.
- FS features that are references to FS excluded from the projection will get a unique value that will be used to generate an exception to service code attempting to instantiate that reference.

3. Delta CAS capability will be built into the XMI serialization done in a service, and into the deserialization done on the client. Deserialization already supports merging new content into an existing CAS; a new deserialization option will enable or disable modifications of preexisting CAS content. Note that the XMI standard already includes add, delete and replace elements to be used in the delta CAS format.

The current XMI CAS serialization format is used to represent the Delta CAS serialzation with the addition of the following three attributes to the cas::View element:

1. added_members - list of XMI ids of feature structures newly indexed in this view.
2. deleted_members - list of XMI ids of feature structures delete from this view.
3. reindexed_members - list of XMI ids of feature structures that need to be reindexed as the feature values may have been modified.