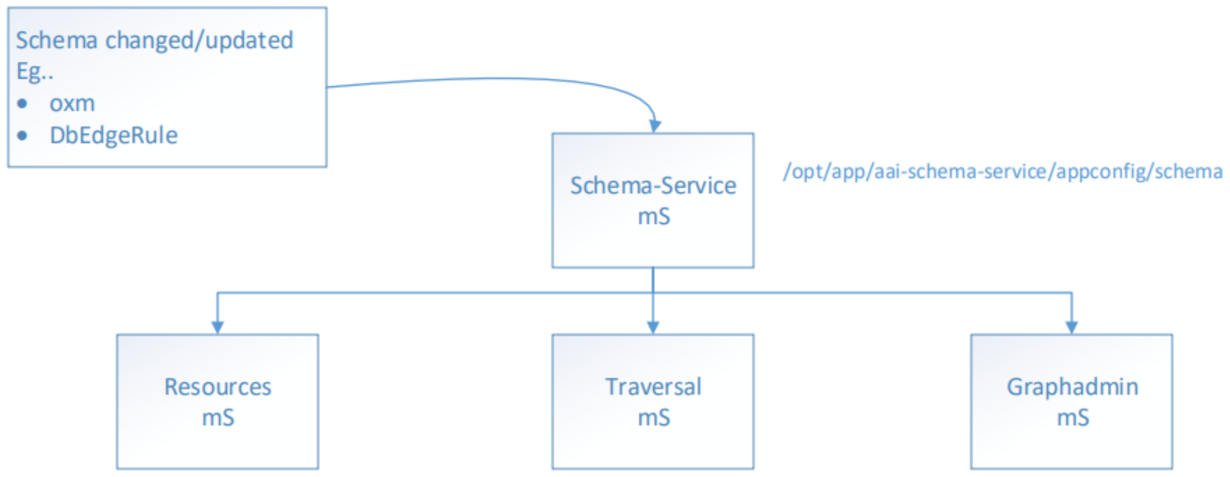
AAI Schema Service

Contributors:

Harish Kajur, Robby Maharajh, William Reehil, CT Paterson, Steven Blimkie

mS Details



Upon startup, these api s (below**) are used to get the latest schema from Schema-Service

- There are 2 options whether to use the schema from Schema-Service or from local schema directory - at /opt/app/aai-resources (aai-traversal or aai-graphadmin)/appconfig/application.properties - schema.translator.list=schema-service OR schema.translator.list=config · If = schema-service, get latest schema from SS (only when starting up)
- If = config, use existing schema from local directory - /opt/app/aai-resources (aai-traversal or aai-graphadmin)/appconfig/schema (api should be working fine even if SS is down) **** There is no guarantee that schema version at local directory will be the same as in Schema-Service mS or get updated on the fly. A setting schema.translator.list= config should only be used in a very rare situation in case SS can t start up. **Ecomp (AAI-14983)

- /aai/schema-service/v1/nodes?version={version}
- /aai/schema-service/v1/edgerules?version={version}
- /aai/schema-service/v1/versions
- /aai/schema-service/v1/stored-queries

New Installation Requirement

- Schema-Service has to be installed and started (in healthy status) before other mS.
- If there is a new Schema-Service build installed; Resources, Traversal and Graphadmin mS needs to be restarted
- Resources, Traversal and Graphadmin will fail to restart if Schema-Service is not up running (unless schema.translator.list=config

----- Everything Below was during the design/requirements phase of schema service ----------

Overview:

- Currently, changes in schema are delivered via new builds of microservices that consume them
- Tracking and testing changes via new builds takes longer

Dublin Requirements:

1. The solution must provide the foundational layer for consuming dynamic schema changes in the future
2. AAI schema service must support the ability to centrally persist (build-time) and serve (run-time) schema via REST
3. AAI schema service must support the ability to centrally persist (build-time) and serve (run-time) custom queries via REST - removing the custom query definitions from the traversal service and making the schema service responsible for them.
4. AAI schema service must support the ability to provide a complete schema as one document even if persisted via multiple files
5. AAI schema service must support the ability to provide a list of documents stored
6. AAI schema service must support the ability to provide an individual document
7. AAI schema service must support the ability to provide associations/grouping between documents *(needs more clarity)*
    a. OXM and Edgerules paired by version [v11, v12, v13]
    b. Grouped by usecase w/ multiple OXM files
8. AAI schema service must continue to support clients that consume the schema via XSDs and POJOs as build-time artifacts
    a. MSO uses this, and would have to update configuration for the location
    b. Followup item: Check with SEs on who is consuming the XSDs
9. Client AAI microservices that are configured to depend on the Schema Service will wait for the AAI Schema Service instance to start
10. AAI must support the option to load the schema files in a development mode (loading locally)
11. Client AAI microservices that currently depend on aai-common / aai-schema artifact at build time must use the AAI schema service REST API as its source for OXM schema files and edge rules (if configured to do so)
12. Epic for effort: ~~AAI-1859~~ - Schema Abstraction CLOSED

Use cases/open questions:

1. Multiple instances of AAI running with entirely separate schemas
    a. William Reehil - The user would specify which schema it requires when communicating with the schema service regardless of version or instantiation of the framework (implementation details will be worked out)
2. Scenario where an attribute is changed interactively by the user - system engineer or dev for testing/proof of concept - and then how to commit that change to the schema service to make it permanent
    a. William Reehil - For Dublin, schema changes would still go through version control. Future enhancements would include the ability to make changes dynamically at run-time, as long as the changes are non-disruptive.
3. Differences between build-time and run-time dependencies - current function, user updates schema file(s) and/or edge rules. This is a run-time interface to a static configuration. Even though it can be set/changed at run-time, for Dublin, the schema/edge rules will remain a build-time dependency for the schema service but is a run-time dependency for the internal AAI microservices that consume the schema service.
    a. William Reehil - Schema is not targeted to be modified at runtime for Dublin, future enhancements.
4. AAI clients that use XSD will consume the schema service artifacts, and will not consume an XSD via REST API
5. What happens when we have conflicts - what conflicts can arise and how they detected and remediated
    a. Auditing on multi-oxm overlap / conflicts
    b. Multi-oxm to split administrative types from inventory types
    c. Does the mutli-oxm which allows the same java-type in more than one file make sense?
6. Impact on history feature - follow up with William Reehil and Robby Maharajh, invite them to next week's call to discuss
    a. William Reehil - Only impacts Chameleon & Champ, changing how they retrieve the schema information, they would now retrieve from the schema service. How Champ loosens it's constraints is up for discussion.
    b. William Reehil - Schema migration changes will be reflected as just changes in the history of the node, there could be special cases here, but we should not allow dynamic changes to the schema that require schema migrations. Note - history design is not finalized since it had been put on hiatus during the design phase.
7. Sparky, Resources, Traversal, GraphAdmin, GraphGraph, Champ, Chameleon will use the schema service to get knowledge into the schema and edge rules of the instance.
8. **Vision:** The schema service will allow various ways to get information about the schema and edge rules through its exposed endpoints, allowing for scope control and ease of understanding. Consumers should not be bound to a singular file format or consummation of the entire configuration file. This is increasingly important when speed is essential and the format should be flexible to accommodate its users. For example, say Sparky just needs to know the attributes of the x node, it should not have to load the entire configuration file in XML format to find out that information. Ideally it should just be able to ask the schema service "What are the attributes of the x node in JSON format?" Then taking this a step further we would like to leverage this service to act as the central hub of schema modifications, but this will require in-depth discussions on how to make this work.

Hey William Reehil (it's me Pavel Paroulek) - I think you written exactly what I suspected - that it would be a mistake to start the implementation of schema service In Dublin, the only selling point which was mentioned by Manisha Aggarwal (dynamic changes, model driven services) is not needed and will be in the "future enhacements". Having a webservice for static content (which as you mention would remain "the schema/edge rules will remain a build-time dependency") doesn't make any sense. Respect +1 🙂. One thing though - I think your technical argument that it's faster to resolve information about the schema through REST than through direct parsing and memory lookup is not correct (but we can discuss it on the meeting).

I think we should now concentrate on the real issue which Venkata Harish Kajur raised - the problematic build times, ease of change and AAI common dependency. I agree with Harishs assessment. We can discuss this all on the meeting.

Proposed Design :

1. A mS called Schema Service will hold and load all the schema (OXM and edge rules) at start up
2. The Schema Service will hold and load the custom query document at start up
3. The Schema Service will provide REST endpoints via GETs such as /aai/schema-service/{api-version}. Consumer mS, such as resource, traversal should use the updated ingest library to make REST calls to the Schema Service
4. The REST endpoint for providing the schema will support a "format" query parameter which will describe what format it is requested in eg. GET /aai/schema-service/v1/nodes?version={version}&format=OXM
5. The REST endpoints such as /aai/schema-service/v1/nodes?version={version} and /aai/schema-service/v1/edgerules?version={version} will provide a complete schema from multiple files
6. The REST endpoint such as /aai/schema-service/v1/list will provide the list of documents stored

7. The REST endpoint such as /aai/schema-service/v1/nodes?version={version}, /aai/schema-service/v1/edgerules?version={version} and /aai /schema-service/v1/stored-query will provide the individual document
8. The REST endpoint will provide the associations between documents via query parameter "version"
9. The schema jar will continue to be provided as an artifact for consumers of XSD and POJOs
10. HEAT environment and OOM environment will need to be addressed for Requirement #9

Future enhancements:

1. AAI should support the ability to consume new schema dynamically, without downtime (eg. when distributed by SDC)
2. AAI should support the ability to notify consumers of schema when new updates are available
3. AAI should support an interface to validate proposed schema changes
4. AAI should support the ability to provide the schema via flexible document formats (OXM, TOSCA etc.)

#1 and #2 need more analysis of use cases and design. #3 seems to align better with the GraphGraph proposal for viewing and interfacing with the schema. Manisha Aggarwal I think it makes sense for all of what was in graphgraph core to be absorbed by Schema Service, when GraphGraph is being worked.