1 The Physical View

2 The Physical View

The world is flat. Ever was. The same goes for TextGrid. Hierarchies are only built by the application logic of the upper levels of the TextGrid infrastructure, primarily by the TextGrid Lab:

- The Projects are modelled according to information extracted from the rights management system (OpenRBAC / LDAP)
- Aggregation Objects (generic ones as well as Editions and Collections) have no content except a list of URIs of the “contained” objects

Out of this information the Navigator builds its tree view.

Physically, on (grid) file system level, TextGrid objects consist of file pairs (tuples), one file for the metadata and one for the object itself (resp. the content), such as:

```
textgrid:1234ab.1.meta
textgrid:1234ab.1
```

This brings us to the identifiers used in TextGrid. TextGrid-URIs are NOIDs with a textgrid: prefix and an ascending numerical suffix indicating the revision number. A TextGrid object is identified and referenced (metadata and elsewhere) by its URI e.g. – to stick with the example: textgrid:1234ab.1 for the first, textgrid:1234ab.2 for the second revision and so on. Per default, the Navigator displays only the latest revision of an object. The latest revision of an object can also be referenced by a logical URI without suffix: If textgrid:1234ab.2 the newest/latest revision of an object, calling or referencing textgrid:1234ab will produce the same physical object. This concept is similar to the eSci-Doc solution – except the underlying terminology (TextGrid: Revision, eSciDoc: Version) and the fact that creating a new revision in TextGrid is a deliberate action to be performed by the user while in eSciDoc versions are generated automatically with every update of an object.

A possible use case for using logical URIs is to use them as references in a Collection Object, so that this Collection contains always the latest version of the respective objects. As every Aggregation, the content of a Collection Object consists only of a list of the URIs of the con- tained objects – formatted in RDF as a very simple Resource Map according to the OAI-ORE specification:

```
<rdf:RDF xmlns:ore="http://www.openarchives.org/ore/terms/" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#">
        <rdf:Description xmlns:tei="http://www.tei-c.org/ns/1.0" rdf:about="textgrid:26dt.1">
                <ore:aggregates rdf:resource="textgrid:21sn" />
                <ore:aggregates rdf:resource="textgrid:21sp" />
                <ore:aggregates rdf:resource="textgrid:21sq" />
                <ore:aggregates rdf:resource="textgrid:21sr" />
                <ore:aggregates rdf:resource="textgrid:21ss" />
                <ore:aggregates rdf:resource="textgrid:21st" />
                ...
                ...
                <ore:aggregates rdf:resource="textgrid:21w4" />
                <ore:aggregates rdf:resource="textgrid:21w5" />
                <ore:aggregates rdf:resource="textgrid:21w6" />
        </rdf:Description>
</rdf:RDF>
```

2.0.0.1 "External" Objects

There is also a mechanism to include objects in TextGrid that are stored or „hosted outside the TextGrid Rep. It is possible to perform a metadata-only ingest with metadata provided by or harvested from other repositories. In such a case, TextGrid includes the URL of this object in the (TextGrid-)metadata and creates an otherwise empty object. In this way, TextGrid users are enabled to work with „external objects for instance by embedding them in Collections or perform queries across the metadata of both „internal and „external objects.

(insert image here)

Physical Type
Metadata Type
Aggregation
Item Edition Collection
(Simple) Object
Item
Ø
W ork Item (External Object)