SEVENTH FRAMEWORK PROGRAMME

CAPACITIES

Research Infrastructures

INFRA-2007-1.2.1 Research Infrastructures

DRIVER II

Grant Agreement 212147

"Digital Repository Infrastructure Vision for European Research II"

Aggregation Advancement Specification

Deliverable Code: D8.2

Document Description

Project	
Title:	DRIVER, Digital Repository Infrastructure Vision for European Research II
Start date:	1st December 2007
Call/Instrument:	INFRA-2007-1.2.1
Grant Agreement:	**212147**

Document	
Deliverable number:	D8.2
Deliverable title:	Aggregation Advancement Specification
Contractual Date of Delivery:	31 st of May 2008
Actual Date of Delivery:	15 th of July 2008
Editor(s):	UNIBI, CNR
Author(s):	Jochen Schirrwagen, Marek Imialek, Paolo Manghi, Marko Mikulicic
Reviewer(s):	
Participant(s):	
Workpackage:	WP8
Workpackage title:	Enhancement of Data Layer-Services
Workpackage leader:	UNIBI
Workpackage participants:	CNR, UGOE, SURF, NKUA, UNIBI
Distribution:	Public
Nature:	Deliverable
Version/Revision:	1.0
Draft/Final:	Draft
Total number of pages: (including cover)	13
File name:	D8.2.pdf
Key words:	Harvesting, Aggregation

Disclaimer

Table of Contents

Table of Figures

Summary

This document describes changes and enhancements to the existing aggregation process in the DRIVER Testbed in order to widen the import to other metadata formats beside Dublin Core (DC), to ensure their quality and to prepare affected services for the aggregation of compound objects. This specification sets the functional context, the detailed technical specification for the development process is maintained and dynamically developed in the virtual collaboration tools of the project (e.g. WIKI).

1 Introduction

Within the DRIVER-I project an aggregating service, namely the Aggregator Service, has been developed and proofed in the DRIVER-Testbed. This service enables the harvesting of Dublin Core metadata from selected repositories of European countries and the normalization, cleaning and enrichment of these metadata to transform it into the DRIVER Metadata Format (DMF).

The objectives of DRIVER II – as described in the DoW[2] – go beyond that. In a consolidation phase the quality of data aggregated from repositories will be improved by help of validation and feedback services and in accordance to the DRIVER Guidelines. In an enhancement and extension phase the harvesting of additional metadata formats will be supported.This is one precondition for the aggregation of compound objects.

To overcome several limitations of the current aggregator service, as they can be concluded from operating the DRIVER-Testbed, a redesign will be necessary. This will lead to a separation of the main operations:

- harvesting metadata formats from OAI-PMH repositories external to DRIVER;
- transforming metadata format into DRIVER into other formats.

This document describes the new Services required for this separation.

This specification sets the functional context for the respective developments in work package 8 (Enhancements of the Data Layer Services) leading to a final release in month 18. The detailed technical specification for the development process is maintained and dynamically developed in the virtual collaboration tools of the project (e.g. WIKI [1]).

2 Enhancement of the Aggregation process

2.1 Aggregation Process Flow

The DRIVER - infrastructure consists of three service-layers – enabling layer, data layer and functionality layer.

The Manager Service, which belongs to the enabling layer, is capable of orchestrating other services by assigning tasks to them and keeping track of their progress. More than one Manager Service can exist, capable of orchestrating certain services in order to achieve application specific conditions and functionalities. The aggregation process flow is the result of an orchestration of services of the data layer, namely for harvesting (Harvesting Service), normalizing, cleaning and enriching (Transformation Service), in combination with validating the metadata (Validator Service), storing and indexing them (MDStore and Index-Service). The two orchestrating services are the Harvesting Manager Service, in control of the available Harvesting Services so as to respect the harvesting operation schedules; and the Transformator Manager Service, in control of the Transformation Services available so as to respect the transformation activities schedules.

The general process flow for the DRIVER Information Space application consists of the following phases and steps:

Enabling Repositories for harvesting (Repository Manager Service)

(1) initial Repository Registration

(2) initial Repository Validation and enabling the repository if validation succeeded

Setting up harvesting and DMF transformation for enabled repositories (DRIVER Manager Service)

(3) creation of Harvesting Instance Data Structure for repository: assignment of source repository to newly created target MDStore Data Structures (one for each metadata format to be harvested) and setting harvesting schedule and typology (refresh or incremental)

(4) creation of Transformation Instance Data Structure for repository: configuration of source MDStore DSs (the ones from the repository) and newly created target MDStore DSs (for DMF records) transformation schedule (refresh or incremental)

Harvesting activities (Harvesting Manager Service)

(5) the manager orchestrates the Harvesting Services available to satisfy the requirements specified by the Harvesting Instance DSs available in the system: Repositories are harvested at the given time intervals, according to the harvesting mode specified, and their records are stored into the specified MDStores

Transformation activities (Transformation Manager Service)

(6) the manager orchestrates the Transformation Services available to satisfy the requirements specified by the “activated” Transformation Instance DSs available in the system. Note that Transformation Instance DSs become “active” once an Aggregator Manager (admin user) has specified the mapping rules to be applied in the transformation process. The manager chooses an available Transformation Service and applies the mapping at the given time intervals, according to the rules specified, and storing the resulting records in the specified MDStores

The aggregation process will be enhanced in its ability to harvest additional metadata formats and running multiple instances of the harvesting service.

Furthermore by providing the necessary protocols, like OAI-PMH through the harvester service interface other consumer services do not need to re-implement its own harvesting protocol.

3 Improving the Services

To achieve the above mentioned enhancements, the Aggregator-Service[3] used in the DRIVER-Testbed needs to be decoupled and separated into different services, namely Harvesting Service and Transformation Service. These services are then orchestrated to form an aggregation process by the Manager Service.

Furthermore the services have to communicate by the subscription-notification mechanism as the DRIVER way of inter-service communication. Due to the fact that harvesting and also transformation operations might have a long processing time, the implementation of callback routines are important.

That way the progress and potential problems during these operations can be monitored by the Manager Service and traced by the Aggregation Manager.

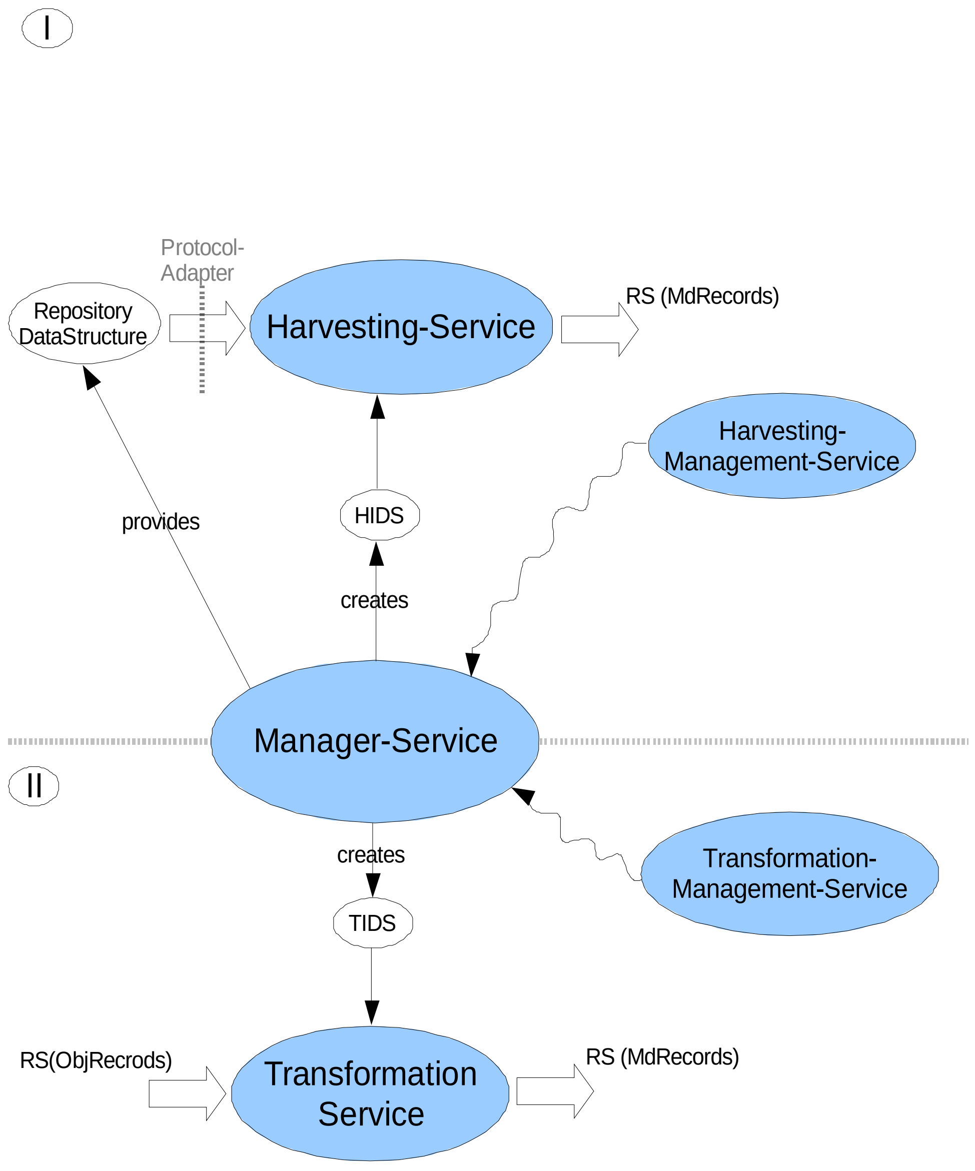

Figure 1 Organizing the Aggregation Process

3.1 Repository Management Service

The Repository Management Service manages Repository Data Structures (RDSs), which represent the repositories registered to the infrastructure with a profile containing information about the OAI-PMH interfaces of the repository (URI, formats, OAI-Sets) and the provenance (administrator, web site, country, etc). The service delegates OAI-PMH standard calls to an available Harvesting Service. Unlike OAI-PMH, it will offer extra methods for harvesting more than one format at a time and return Object Records.

3.2 Harvesting activities

The *Harvesting Service* replies to OAI-PMH calls for a given Repository Data Structure. Depending on the OAI-PMH method, the service returns on:

- listRecords: a resultset containing all records. which meet the harvesting criteria metadataformat and optionally set and datestamp;
- getRecord: an individual record;
- listSets: a resultset with the OAI-sets as strings;
- listMetadataformat: a resultset with the available metadataformats

The harvested result will be returned as Driver Metadata Records into a resultset. In the case of the Harvesting Manager Service – asking the Harvesting Service to perform a given operation – will then take the resultset and feed it to an MDStore. The Service maintains logs of the harvesting activities for each Repository Data Structure. Logs can be requested on demand by other services through appropriate management methods, by specifying the given Repository Data Structure.

The *Harvesting Manager Service* provides other consumer services a facade to the Harvesting Instance Data Structure. Additionally it provides a User-Interface, in which Aggregator Managers specify a number of parameters for the harvesting operation, like metadata formats, time scheduling and the Harvesting Protocol Adapter to be used.

The representation of logs gives information on harvesting progress and potential errors.

The *Harvesting Instance Data Structure* is a resource which contains informations about the source repository, the target MDStore-DS, the OAI settings and the scheduling.

Additional protocols for harvesting operations may supported in the future.

3.3 Transformation Activities

The Transformation Service provides operations on valid XML-Documents.

It accepts resultsets of object records and Transformation Instance Data Structure. As a result, it returns a resultset with the DRIVER metadata records obtained by applying the transformation rules of the Transformation Instance Data Structure.

Additionally the Transformation service supports logging of the performed actions.

The *Transformation Manager Service* provides a User-Interface for the definition of transformation rules, the storage of predefined set of rules and their application in a specific TIDS.

Transformation rules are defined by a formal transformation language, which allows the definition of scripts that calculate output fields by processing input fields.

Input/output formats of a transformation are specified by two MDFormat Data Stuctures and their respective layouts.

This approach will allow the transformation of records from any (flat) input format into any (flat) output format.

The *Transformation Instance Data Structure (TIDS)* contains information about the source and target of the mapping, the transformation rules to be applied and the scheduling.

3.3.1 Feature Extraction Service

The “Feature Extraction Service” belongs to the category of Utility Services.

It will expect a ResultSet with links of Simple Digital Objects as input and delivers a ResultSet with descriptive MetadataRecords.

Specific algorithms will be used to provide a number of operations on DRIVER objects, such as

- Language Detection
- Keyword Extraction from full texts
- Text extraction from PDF-files and scanned images of textual records e.g. for full-indexing

4 References

[1] ScrewDRIVER Wiki, http://technical.wiki.driver.research-infrastructures.eu

[2] DRIVER Annex I - “Description of Work”, Proposal no. 212147.

[3] Aggregator Service Implementation, http://technical.wiki.driver.research-infrastructures.eu/index.php/Main_documentation_AggregatorService-1.1.0