VOSpace Backend Service Developers Guide 1.0

oats_vospace-backend_devel_guide

INAF-OATs Technical Report 238

Sara Bertocco

Giuliano Taffoni

VOSpace Backend Service Developers Guide 1.0 oats_vospace-backend_devel_guide
INAF-OATs Technical Report 238
Edition 1

Author	Sara Bertocco	*bertocco@oats.inaf.it*
Author	Giuliano Taffoni	*taffoni@oats.inaf.it*

This guide gives software description and some development hints to meke easier to developers to add new storage solutions to their VOSpace service.

Introduction

The *"VOSpace, Version 2.1"*[1] says: "A VOSpace web service is an access point for a distributed storage network. Through this access point, a client can:

- add or delete data objects
- manipulate metadata for the data objects
- obtain URIs through which the content of the data objects can be accessed

VOSpace does not define how the data is stored or transferred, only the control messages to gain access. Thus, the VOSpace interface can readily be added to an existing storage system. When we speak of “a VOSpace”, we mean the arrangement of data accessible through one particular VOSpace service."

The vospace-backend is a storage service management to integrate a VOSpace interface implementation with a storage solution. This software module persists the information linking a VOSpace Node Identifier with a vospace backend stored file, so it is able to return URIs to access the content of the data objects.

The storage solution can be changed thanks to a plug-in architecture allowing to plug-in at run time different storage solutions simply implementing an abstract java class.

The service requires an SQL Resource DataBase Management Service (RDBMS) as default built-in persistence layer.

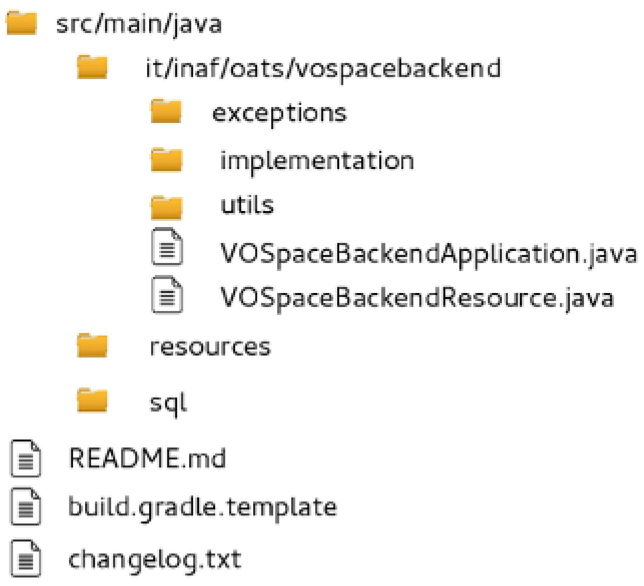
vospace-backend provides a RESTful interface.

[1] http://www.ivoa.net/documents/VOSpace/20150601/VOSpace.pdf

Software description

This software implements CADC interface ca/nrc/cadc/vos/server/transfers/TransferGenerator.java which means substantially imlements the function List<URL> getURLs(VOSURI target, Protocol protocol, View view, Job job, List<Parameter> additionalParams) throws FileNotFoundException, TransientException; the function return, in the most generale case, a list of URLs to access data for the given transfer request information. In the current implementation, a single URL is returned built as Service-base-URL-/OSURI of data to access/signature/jobid

2.1. Development tree



```
src/main/java
    it/inaf/oats/vospacebackend
        exceptions
        implementation
        utils
        VOSpaceBackendApplication.java
        VOSpaceBackendResource.java
    resources
    sql
README.md
build.gradle.template
changelog.txt
```

2.2. Plug-in architecture

The swap between different storage base implementations is realized using a plugin factory.

Steps to realize a plugin factory:

1. define a Java interface for the methods whose implementation has to be swapped.

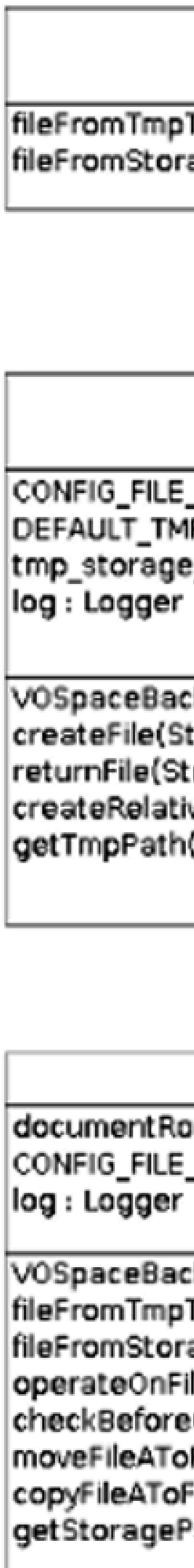
   The oats-vospace-backend implementation minimizes the code to write to add a new storage method using an interface with only two functions: one to store the data and one to retrieve the data. All the common management work is performed in an abstract class implementing the public interface. A new storage supporting object must implement the interface and extend the abstract class.

2. define the concrete implementation(s) of that interface.

   In the current implementation of oats-vospace-backend is available
   a file system posix based storage implemented in the class
   `it.inaf.oats.vospacebackend.implementation.VOSpaceBackPosix`

3. create a plugin factory class with a method to return one of the concrete implementations, as defined by a configuration setting.

   In oats-vospace-backend, the configuration is defined in the file **`VOSpace.properties`** where the parameter **`it.inaf.oats.vospacebackend.implementation.VOSpaceBackendImpl`** defines the concrete inmlementation to be used (for example **`it.inaf.oats.vospacebackend.implementation.VOSpaceBackPosix`**). The file must be located in the subdirectory **`config`** of the home directory of the user running the service. The plugin factory retuns the concrete implementation configured in **`VOSpace.properties`**.


fileFromTmpT
fileFromStora
CONFIG_FILE_
DEFAULT_TMI
tmp_storage
log : Logger
VOSpaceBac
createFile(St
returnFile(St
createRelativ
getTmpPath(
documentRo
CONFIG_FILE_
log : Logger
VOSpaceBac
fileFromTmpT
fileFromStora
operateOnFil
checkBefore
moveFileAToI
copyFileAToF
getStorageP

VOSpaceBackendImplFactory
CONFIG_FILE_NAME : String
<<creates>>

2.3. General functionality description

This paragraph is a general description of the functionality of the i vospace-backend software module.

It is a web service, based on the RESTful architecture, realized using the *Restlet Framework*[1] At present, the VOSpaceBackendResource implements the GET and PUT operations only.

The functionality of PUT operation is based on the following steps:

- Reads the request input parameters:
  - "jobid"
  - "parameters" contains the encoded vosuri of the VOSpace Node relative to the operation. This parameter has to be decoded. It is originally encoded for security reasons.
- Downloads the input file associated with the request putting it in a temporary area with a unique id as name and calculates its md5 checksum
- Stores the received file:
  - initializes the Node metadata in the VOSpace interface backend
  - creates an entry in the vospace-backend database to store the relation between the Node identifier and the stored file identifier
  - move the received file from the temporary area to the final storage area. This is the only operation specific of the adopted storage solution
- Updates the job status.

[1] https://restlet.com/open-source

How to add a new storage solution support

The vospace-backend software is designed to be extended to support more storage solutions implementing new plug-ins.

This chapter explains how to write a new plug-in.

3.1. How to write a new plug-in

Reading the architecture, it is clear that a new plug-in is a java class extending VOSpaceBackend.java, which means implementing the two methods of the VOSpaceBackendInterface.java:

```
public void fileFromTmpToFinalStorageArea(String storedFileID, String md5_sum)
public String fileFromStorageAreaToTmp(String vosuri, String storedFileID)
```

Appendix A. Revision History

Revision 1.0-0 Fri March 03 2017 **Sara Bertocco**
sara.bertocco@oats.inaf.it

Initial creation by publican

Index