




# Final Platform Release

## Authors

Rainer Schmidt, Matthias Rella (AIT Austrian Institute of Technology)

January 2014

*This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137).*

*This work is licensed under a CC-BY-SA International License* 



## Executive Summary

The SCAPE Platform Sub-project has released a virtual machine image allowing users to easily execute preservation tools in a parallel execution environment based on Apache Hadoop. This virtual machine-based release provides a low barrier of entry for new users to experiment with the basic SCAPE platform, which has already been utilized in the context of the SCAPE training events. This deliverable comprises a virtual machine image which comes with a set of training examples and corresponding documentation.

The release provides an experimental environment that combines Taverna Workbench, the Platform's MapReduce tool executor (ToMaR), a set of preservation tools and workflows, which can be easily run on a desktop computer. The virtual machine image provides also a well configured environment that can be extended with additional Platform components and/or be deployed on a parallel hardware infrastructure. This document provides an overview of the virtual machine deployment, the utilized software, and example workflows.

## Table of Contents

1.	Introduction .....	1
2.	Contents of the Virtual Machine Image.....	2
2.1.	Structure on the Filesystem .....	2
2.2.	Provided Example.....	3
3.	Using OpenJPEG and Jpylyzer .....	4
3.1.	OpenJPEG .....	4
3.2.	Jpylyzer.....	4
4.	Tavena-based Preservation Workflows .....	5
5.	Scalable Preservation Tools with ToMaR and Taverna.....	7
6.	References.....	10
<b>Appendix A Instructions for Using Hadoop on the Virtual Machine Image .....</b>		<b>11</b>
	About .....	11
	Starting the Virtual Machine .....	11
	Logging into the Virtual Machine .....	11
	Starting Hadoop.....	11
	Run your first Job.....	12
	Monitoring HDFS/Jobs.....	12
	Hadoop Command Reference .....	13
	Using the Guidelines.....	13
<b>Appendix B Instructions for Tiff2JP2-Example.....</b>		<b>13</b>
	Integrating Taverna with Hadoop .....	13
	Performing the task from the command-line .....	13
	Implementing the Taverna workflow .....	13
	Using the SCAPE MapReduce Tool Wrapper (ToMaR) .....	14
	Using Taverna to orchestrate Hadoop .....	14
	Workflow + Image Processing .....	15
<b>Appendix C Instructions for Cluster Setup.....</b>		<b>16</b>
	Desktop Setup.....	16
	Using Multiple Cores .....	16
	Cluster Setup .....	16



## 1. Introduction

The primary goal of the SCAPE Platform is to enhance the scalability of preservation activities. In order to support this vision, SCAPE is applying data-intensive computing technologies in the digital preservation domain. This involves the employment of a scalable architecture and technologies for parallel processing, automated data decomposition, and error recovery. The preservation environment delivered by the SCAPE platform builds on top of a set of frameworks like Apache Hadoop, Taverna or the presently developed Fedora 4 repository, as previously discussed in [1].

A major hurdle for users that intend to experiment with the SCAPE Platform is the complex setup of the system. Various Platform components may be used stand-alone or combined with other components. The execution system can be configured to execute workflows on a desktop computer, a remotely hosted service and/or a computer cluster. The flexible design has the benefit that one can set up a SCAPE platform in many different environments. It has also turned out that individual customization of the deployments utilized by the SCAPE content holders was needed in order to take advantage of the different institutional settings and to implement the respective use-cases in an efficient and scalable fashion.

In order to help SCAPE users to experiment with the basic framework and tools used in the context of the SCAPE platform, SCAPE has released a pre-configured virtual machine image. The idea of this release is to provide a basic environment that can be easily installed on a desktop computer allowing users to experiment with different technologies and examples. The virtual machine release may also be deployed as a cluster on multiple machines and be extended with additional SCAPE software components.

This document summarizes the content of the SCAPE Platform virtual machine release, which has already been used for hands-on exercises carried out during the SCAPE Training event held in September 2013 at the British Library in London<sup>1</sup>. Besides the software deployment, the release also contains a set of examples and tutorials provided by Rainer Schmidt (AIT), Matthias Rella (AIT), Sven Schlarb (ONB), and Carl Wilson (OPF). The release is presently available as an image for VirtualBox<sup>2</sup> which utilizes the Ubuntu 10.04.4 LTS<sup>3</sup> Linux distribution as OS<sup>4</sup>. The image is presently available as a downloadable package to SCAPE participants and be made publicly available once licensing restrictions have been clarified.

---

<sup>1</sup> <http://wiki.opf-labs.org/display/SP/SCAPE+Future+Formats+First+Agenda>

<sup>2</sup> <https://www.virtualbox.org/>

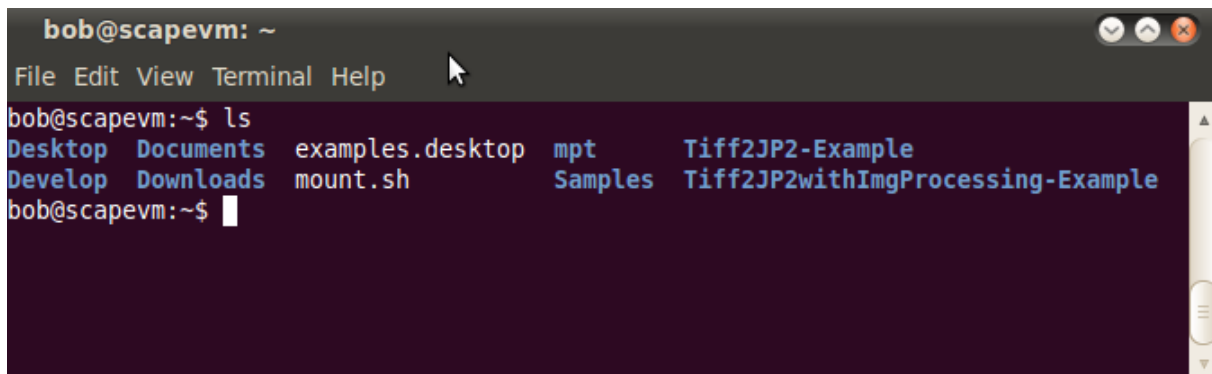
<sup>3</sup> <https://wiki.ubuntu.com/LTS>

<sup>4</sup> We are planning an upgrade of the VM image to the latest Ubuntu LTS release together with an update of the SCAPE platform components for the final project months.

## 2. Contents of the Virtual Machine Image

### 2.1. Structure on the File System

The released virtual machine is based on the Ubuntu 10.04 LTS Linux distribution and requires the Oracle VM VirtualBox hypervisor to be run on a desktop environment. The recommended VirtualBox version is 4.2.16. The virtual machine image (VM) is configured in a way that all required software can be used from the default Linux user account “bob”. The examples are configured to use Apache Hadoop 1.2.1 as the parallel execution environment, Taverna Workbench as the workflow engine, and the Platform’s tool executor (ToMaR) as the underlying MapReduce application. All three software components are Java-based and deployed under user permissions in the user’s home directory. The VM image contains also a set of preservation tools which have been installed based on individual installation routines. The default user account can be used to execute commands under super-user privileges using the Linux command ‘sudo’. This allows one to install additional preservation tools as for example those Debian packages created in the context of the SCAPE Preservation Components (PC) sub-project.

A screenshot of a terminal window titled "bob@scapevm: ~". The terminal shows the output of the command "ls" in the home directory. The output is as follows:

```
bob@scapevm:~$ ls
Desktop  Documents  examples.desktop  mpt      Tiff2JP2-Example
Develop  Downloads  mount.sh          Samples  Tiff2JP2withImgProcessing-Example
bob@scapevm:~$
```

Figure 1 – Home directory for the default user account

Figure 1 shows the folder structure of the default user’s home directory (/home/bob):

- The “Develop” folder contains the pre-installed Platform components used by the release. This includes JDL version 1.6, Taverna workbench 2.30, Apache Hadoop version 1.2.1, and Oozie version 3.2.2. For the provided examples, Oozie can be optionally used as a Hadoop Rest interface and job scheduler. In the default set-up Hadoop is however directly invoked via the command-line interface (optionally via an SSH connection). “jce” provides the ‘Unlimited Strength Java Cryptography Extension’ policy for Java, which is used by Taverna for confidential message exchange [2].
- The “Downloads” folder contains the software packages that are required to run the examples and that have been pre-installed on the VM image. This includes Apache Hadoop v. 1.2.1, Taverna Workbench v. 2.3.0, Java JCE policy 6, the SCAPE cipex and cipex-analyse<sup>5</sup> tools for identifying files packaged in container filers. The folder also includes a set of image processing and preservation tools (including FITs v. 0.6.1, jpylyzer<sup>6</sup> v. 1.6.0 + jpylyzer test

<sup>5</sup> <https://github.com/shsdev/cipex-analyse>

<sup>6</sup> <http://openplanetsfoundation.org/software/jpylyzer>

files, openjpeg v. 1.5.0 + v. 2.0.0). Other pre-installed tools, which are not used by the basic examples, include: the SCAPE Matchbox tool<sup>7</sup>, Exiftool (libimage-exiftool-perl), SCAPE XMLworkflowReport-1.0 (in /usr/share).

- The “Documentation” folder contains instructions for configuring and running the examples provided by the virtual machine image along with a set of presentations that were given at the SCAPE training event in London in September 2013.
- The “Samples” folder contains sample data sets that can be used as input data for the example workflows pre-installed on the VM. The example files comprise an image test set widely used for image processing, a test set of documents taken from the Govdocs1 corpus, the jpylyzer Jpeg2000 test files, as well as a set of high resolution newspaper scans.
- The folders “Tiff2JP2-Example” and “Tiff2JP2withImgProcessing-Example” contain the set-up for two training examples which will be explained in more detail in the following sections of this document.

## 2.2. Provided Example

The main goal of this virtual machine release is to provide a step-by-step tutorial for practitioners to using basic elements of the SCAPE platform software and to implement custom workflows and solutions using different techniques. Please note that we did not aim at putting all of the SCAPE and/or SCAPE Platform outcomes onto this virtual machine release. The reason for this is that we wanted to keep this environment simple comprising only necessary and widely used components. This enables practitioners to quickly understand the set-up allowing them to create similar deployments within their institutional environments. The set-up comes with almost no inter-dependencies between the deployed components making it very easy to extend the deployment with additional components. Other Platform outcomes like Fedora 4<sup>8</sup> are compliant with the software deployed on this VM and will be put on a later version of this VM, once they have been released.

The examples presented in this release are intended to provide a method allowing one to migrate commonly used workflows from a desktop environment to a Hadoop platform. The use-cases have been taken from the SCAPE Testbed user stories implementing scenarios like identification, file conversion, and quality assurance. The examples make use of commonly used preservation tools which are step-wise enhanced towards supporting automation, parallel, and remote execution. We explain how preservation tools are executed, how their execution can be automated using workflow technology, how they can be run in a scalable fashion using a Hadoop cluster, and how these technologies can be integrated and used to process large volumes of data in a scalable and possibly remote environment.

In order to demonstrate the development of a scalable workflow based on a typical use-case, the virtual machine release provides an example workflow which is available in different versions. This example implements a common use-case which is the migration of digitized images available in the TIFF format to the JPEG2000 format. The folder “Tiff2JP2-Example” contains a set of implementations of this use-case. Starting with a batch script, the workflow is further developed to be executed as a Taverna workflow, to be executed as a MapReduce job, and finally to an integrated Taverna-workflow that orchestrates parallel jobs running on a Hadoop cluster. The example also

---

<sup>7</sup> <http://www.scape-project.eu/leaflets/matchbox-the-duplicate-image-detection-tool>

<sup>8</sup> <https://github.com/futures/fcrepo4>

demonstrates how a SCAPE format validation component (jpylyzer) is added to the migration workflow. The folder “Tiff2JP2withImgProcessing-Example” provides an even more advanced version of the same workflow that implements quality assurance by comparing technical metadata and image processing results. This workflow adds Taverna activities that make use of ExifTool, the ‘extractfeatures’ implemented by the SCAPE Matchbox tool, which are applied to both TIFF and JP2 files. The matchbox ‘compare’ command is then used for SIFT feature comparison and an XML report generator is used to present the obtained results. Instructions for starting the execution platform and for configuring and running the different examples are provided in the “Documents” folder on the virtual machine and in Appendix B of this document. Instructions on how to set-up, test, and monitor a running Hadoop instance on the virtual machine are provided in Appendix A of this document.

### 3. Using OpenJPEG and Jpylyzer

OpenJPEG and Jpylyzer are two example tools that have been pre-installed on the virtual machine image and that are used throughout the presented preservation workflow examples. Introductions on how these tools are used are provided in the “Documents” folder on the virtual machine release. The section provides a brief summary on how the tools are used on the VM and provides pointers to more detailed information.

#### 3.1. OpenJPEG

Image\_to\_j2k is part of the OpenJPEG library installed under the path /usr/local/bin/ on the virtual machine. It takes an image of a certain type as input and converts it to a jpeg2000 file.

Example:

```
image_to_j2k -i INPUT -o OUTPUT -I -p RPCL -n 7 -c  
[256,256],[256,256],[128,128],[128,128],[128,128],[128,128],[128,128] -b 64,64 -r  
320.000,160.000,80.000,40.000,20.000,11.250,7.000,4.600,3.400,2.750,2.400,1.000
```

n ... specifies the number of resolutions contained in the file.

c ... specifies the size

r ... specifies the compression ratio(s)

Please consult the on-line reference manual for further details using the command ‘man image\_to\_j2k’.

#### 3.2. Jpylyzer

Jpylyzer is a JP2 (JPEG 2000 Part 1) image validator and properties extractor. The Jpylyzer test-files<sup>9</sup> can be found in the Samples folder on the virtual machine.

To get the description of the command line arguments use the command

```
jpylyzer -h
```

---

<sup>9</sup> Available at <https://github.com/openplanets/jpylyzer-test-files>

To analyse one of the correct files (e.g. reference.jp2) use command

```
jpylyzer reference.jp2 > output.xml
```

The analysis result is then stored in an XML file. The most important result can be found in the <isValidJP2> tag. If the validation was successful the value is “True” otherwise “False”. To look into the contents of the XML file use an XML editor or Web browser.

The jpylyzer output contains five top level elements:

1. toolInfo - holds information about jpylyzer itself (e.g. version)
2. fileInfo - system level information about analysed image (name, path, size, modification date)
3. isValidJP2 - shows whether analysed document is a valid JP2 document
4. tests - reports information about failed tests; is empty if tests were successful; e.g. colour space is not compatible with JP2 format
5. properties - technical information about the image and follows the box structure of the JP2 format; the meaning of the boxes is documented in the jpylyzer user manual; e.g. the transformation field allows you to distinguish between lossless and lossy encoding

For further information regarding jpylyzer please consult the on-line reference manual using the command ‘man jpylyzer’.

Further examinations on the tool are provided as online video<sup>10</sup>.

## 4. Taverna-based Preservation Workflows

Taverna<sup>11</sup> provides an integral part of the SCAPE Platform providing workflow design, workflow orchestration, and workflow annotation functionality. The fundamental idea is to specify manual tool invocations as a workflow which can be repeated, traced and/or ported to another environment. Taverna workbench provides a graphical environment allowing a user to specify a preservation workflow in form of a workflow document, which can be subsequently executed by the Taverna workflow engine.

The “Documents” folder on the VM image contains a step-by-step tutorial for creating a local preservation workflow. The tutorial describes how the Taverna tool plugin can be used to invoke a pre-installed preservation tool. The same plugin is used to invoke wrapped preservation tools on a cluster by the example use-cases described later in this document. In the tutorial, a workflow is constructed that connects the tools OpenJPEG 2.0, Jpylyzer 1.6.0, FITS 0.6.1, and ImageMagick. Furthermore, it explains the use of Beanshells in Taverna allowing a workflow to iterate through an input text file, utilizing regular expressions, extracting results using Xpath queries, adding control

---

<sup>10</sup> <http://www.openplanetsfoundation.org/software/jpylyzer>

<sup>11</sup> <http://www.taverna.org.uk/>



logic, and handling files. For detailed instructions with code examples and screenshots the reader is referred to the workflow tutorial under /home/bob/Documents on the virtual machine image.

The folder “Tiff2JP2-Example” contains a file tiff2jp2.sh that implements the demonstration use-case as a bash script. The script allows a user to automatically migrate a folder with TIFF images to the JPEG 2000 format, validate the result using Jpylyzer, and display the results. Moreover, the folder contains a workflow document tiff2jp2.t2flow which implements the same workflow in Taverna’s t2flow language.

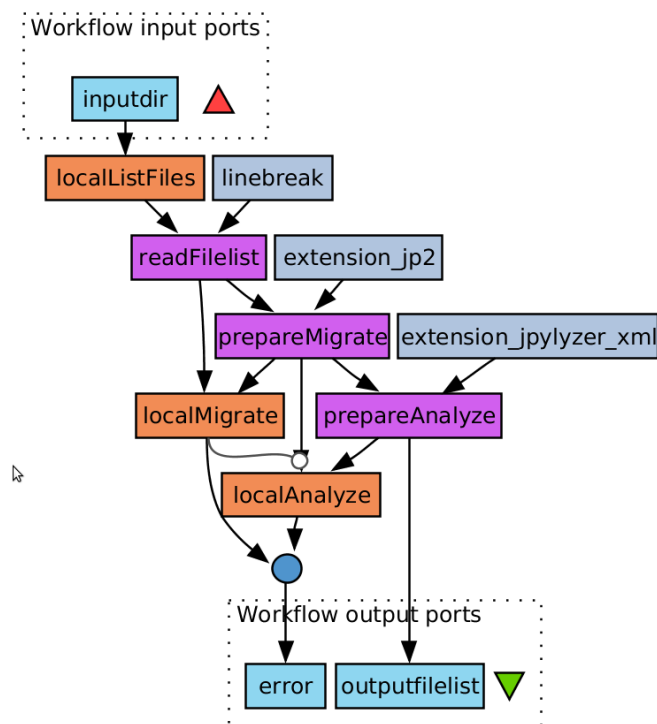


Figure 2 – Local Taverna workflow Tiff2JP2 with Format validation

Figure 2 shows the Taverna workflow for converting TIFF images to the JPEG 2000 format. The orange activity boxes represent invocations using the Taverna tool plugin. The activity “localListFiles” executes a Linux command that generates the list of input files, the activity “localMigrate” invokes OpenJPEG for the file migration, and the activity “localAnalyze” invokes the Jpylyzer file format validator. The purple activities represent beanshell scripts that are used for String operations on the data that is transferred via the input and output ports. The grey boxes represent values like for example a String specifying a file extension (“.jp2”). The blue boxes represent workflow input and output ports. The details of the “localAnalyze” activity which implement a local invocation of the Jpylyzer tool using Taverna’s tool plugin are shown in Figure 3.

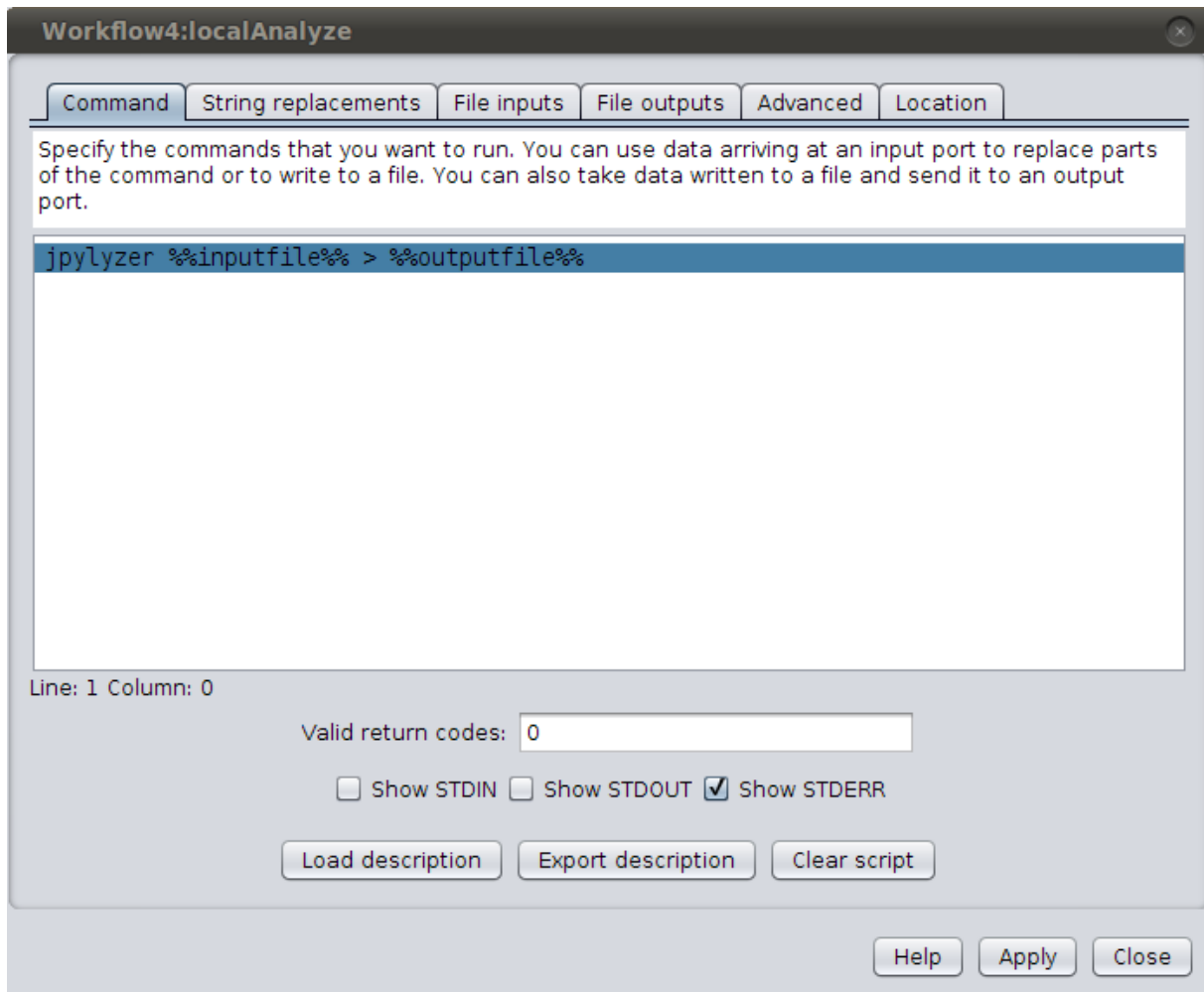


Figure 3 – Using the Taverna tool plugin for invoking a tool locally

## 5. Scalable Preservation Tools with ToMaR and Taverna

The SCAPE Tool-to-MapReduce Wrapper (ToMaR) provides a tool for executing sequential tools in a MapReduce environment. ToMaR can be used to wrap command-line tools or Java APIs for parallel execution on a Hadoop cluster. A typical use-case is the generation of metadata from binary content which can then be further processed using tools provided by the Hadoop ecosystem. A study that uses ToMaR and cipex that extracts metadata from large-volumes of web archive files is described in [[1]].

Cipex is a Java-based tool for identifying files packaged in container files. Cipex can be run locally in sequential mode or take advantage of parallel execution on a MapReduce cluster. Cipex implements the MapReduce programming mode and (unlike most preservation tools) can be directly executed on a Hadoop cluster without requiring an additional wrapper. A step-by-step tutorial on identifying sample Zip container files from the Govdocs1 corpus<sup>12</sup> is provided in the “Documents” folder of the virtual machine distribution. The “Sample” folder contains the corresponding samples from the

<sup>12</sup> <http://digitalcorpora.org/corpora/files>

Govdocs1 corpus. The cipex and cipex-analyse Java archives are available from the “Downloads” folder of the VM release.

The folder “Tiff2JP2-Example” provides an example for wrapping preservation tools with the SCAPE MapReduce tool wrapper (ToMaR). A corresponding step-by-step tutorial can be found in the “Documents” folder of the virtual machine release and in the Appendix of this document. ToMaR is provided as a Java archive (which implements a generic MapReduce application). In order to experiment with the tool wrapper using the Hadoop command-line interface a set of helper scripts are provided. ToMaR makes use of the SCAPE tool specification language<sup>13</sup> for invoking individual tools on the command-line. The folder “Tiff2JP2-Example/toolspecs” contains a set of tool specification documents which are compliant with the pre-installed command-line tools (exiftool.xml, jpylyzer.xml, matchbox.xml, openjpeg.xml, report.xml, zip.xml). The script gen-input.sh provides a convenient way of generating a valid input file required by ToMaR for an arbitrary folder with source images (tiffs in the case of this example). The toolwrapper can then be launched on a Hadoop cluster by simply using its command-line interface (hadoop jar toolwrapper.jar -i input-files.txt -r toolspecs).

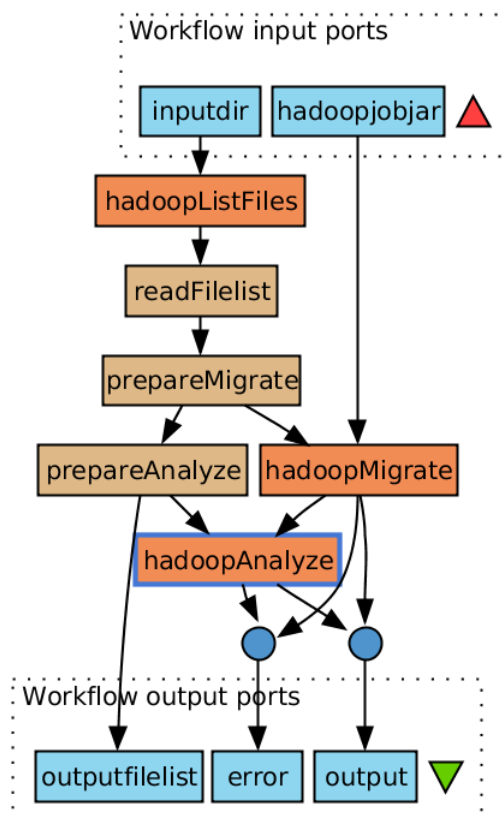


Figure 4 – Extended workflow Tiff2JP2 with Format validation. Tool activities executed in external MapReduce environment.

<sup>13</sup> <https://github.com/openplanets/scape-toolspecs>

The folder “Tiff2JP2-Example” also contains a Taverna workflow document `tiff2jp2_hadoop.t2flow` that uses parallel activities to implement the demonstration use-case, shown in Figure 4. This workflow extends the workflow described in the previous section by utilizing ToMaR in order to launch the preservation activities on the local MapReduce environment. Additionally, the folder contains a `.t2flow` which allows one to launch the demonstration workflow on a remotely hosted Hadoop cluster via an SSH connection. If this workflow is compared to the sequential version shown in Figure 2, one can see that the workflow logic has not significantly changed.

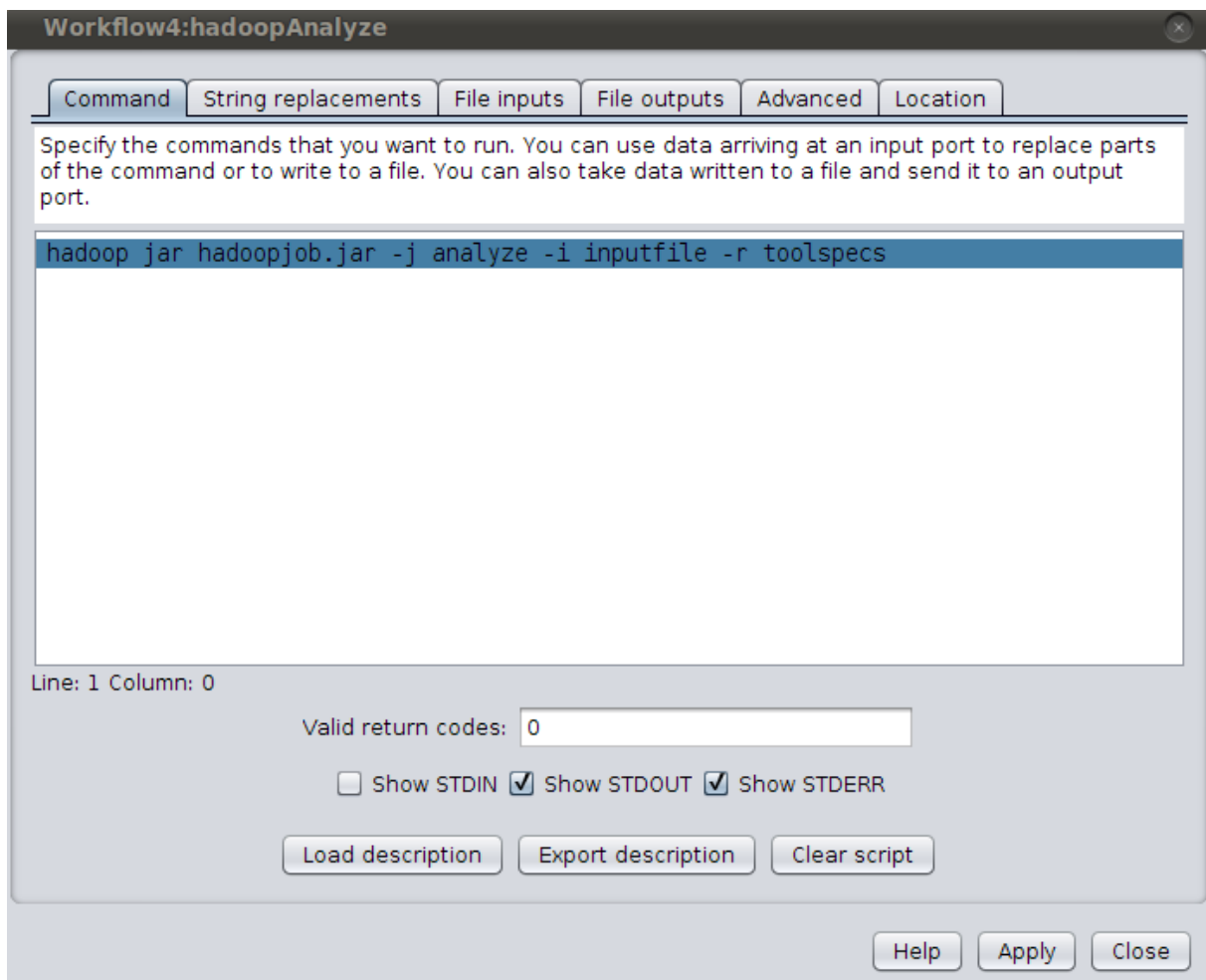



Figure 5 - Using the Taverna tool plugin and ToMaR for invoking preservation tools on a Hadoop cluster

Although the design appears similar, tool invocation and data handling of the parallelized workflow has been fundamentally changed in the parallel version of the example workflow. The Hadoop-enabled workflow consumes an (potentially large) input file, which points to data and operations that will be carried out on the MapReduce cluster. Preservation activities are still implemented using Taverna’s tool invocation plug-in. However, compared to the sequential workflow which uses activities that invoke the preservation tools directly, in the Hadoop-enabled workflows the tool invocation plug-in is used to implement activities that invoke ToMaR.

Figure 5 shows Taverna’s tool plugin dialog as it is configured for the parallel activities. It can be seen that Taverna is now executing a Hadoop command line task similar to the one described earlier, when we explained the usage of the SCAPE MapReduce tool wrapper (ToMaR) from the shell. Please note that the file “hadoopjob.jar” links to the ToMaR version available on the virtual machine release. The “Location” tab provided by the tool plugin dialog is being used to configure the ssh connection when the activity is carried out on a remote execution environment.

 Documentation

Oozie Web Console

Workflow Jobs | Coordinator Jobs | Bundle Jobs | System Info | Instrumentation | Settings

All Jobs | Active Jobs | Done Jobs | Custom Filter ▼

Job Id	Name	Status	Rur	User	Group	Created	Started
1 0000000-130711160035715-oozie-rain-W	TIFF2JPG-seq	SUCC...	0	rainer		Thu, 11 Jul 2013 14:43:08 ...	Thu, 11 Jul 2013 14:43:08 ...
2 0000007-130705121022180-oozie-rain-W	TIFF2JPG-seq	SUCC...	0	rainer		Fri, 05 Jul 2013 14:56:57 G...	Fri, 05 Jul 2013 14:56:57 G...
3 0000006-130705121022180-oozie-rain-W	TIFF2JPG-seq	SUCC...	0	rainer		Fri, 05 Jul 2013 13:03:48 G...	Fri, 05 Jul 2013 13:03:48 G...
4 0000005-130705121022180-oozie-rain-W	TIFF2JPG	SUCC...	0	rainer		Fri, 05 Jul 2013 12:50:48 G...	Fri, 05 Jul 2013 12:50:48 G...
5 0000004-130705121022180-oozie-rain-W	TIFF2JPG	SUCC...	0	rainer		Fri, 05 Jul 2013 12:38:45 G...	Fri, 05 Jul 2013 12:38:45 G...
6 0000003-130705121022180-oozie-rain-W	TIFF2JPG	SUCC...	0	rainer		Fri, 05 Jul 2013 12:19:39 G...	Fri, 05 Jul 2013 12:19:39 G...
7 0000002-130705121022180-oozie-rain-W	TIFF2JPG	KILLED	0	rainer		Fri, 05 Jul 2013 12:14:07 G...	Fri, 05 Jul 2013 12:14:07 G...
8 0000001-130705121022180-oozie-rain-W	TIFF2JPG	KILLED	0	rainer		Fri, 05 Jul 2013 12:00:48 G...	Fri, 05 Jul 2013 12:00:48 G...

Figure 6 – Workflow activities scheduled via Oozie

Apache Oozie which is also available from the “Downloads” folder on the virtual machine release provides a server-sided workflow scheduler and REST interface for Hadoop. If ToMaR-based tool execution is registered with Oozie as a workflow fragment, one can make use of the Oozie HTTP interface to configure ToMaR and schedule its execution. This way, one can also invoke Hadoop-based preservation activities from Taverna using a REST service endpoint, as compared to using the tool-plugin and SSH connections. The Web Console in Figure 6 shows preservation activities which have been scheduled using the Apache Oozie workflow scheduler.

## 6. References

- [1] SCAPE Deliverable D4.1 Architecture Design. Available at: <http://www.scape-project.eu/deliverable/d4-1-architecture-design-first-version>
- [2] <http://www.taverna.org.uk/documentation/faq/security/https-support/>
- [3] <http://www.openplanetsfoundation.org/blogs/2013-12-16-web-archive-fits-characterisation-using-tomar>



## Appendix A Instructions for Using Hadoop on the Virtual Machine Image

### About

The instructions below will allow you to set-up a running Hadoop instance on your local laptop inside a virtual machine. This allows you to easily experiment with Hadoop, execute MapReduce jobs, and test workflows.

### Starting the Virtual Machine

Install the VirtualBox<sup>14</sup> virtualization software package on your laptop. Obtain the SCAPE Platform Virtual Machine files and store them on your laptop. Start VirtualBox, open the SCAPE Virtual Machine, and start the virtual machine.

### Logging into the Virtual Machine

You should now see the login-screen of an Ubuntu Linux installation. Use the username: bob and password: alice as credentials to log into the virtual machine. In order to work with Hadoop, you will need a command-line terminal and a web browser. Use the corresponding icons next to the System menu on the bottom-left corner to start the corresponding applications.

### Starting Hadoop

Execute the following command within a terminal window:

```
bob@scapevm:~$ cd /usr/local/hadoop
bob@scapevm:~$ bin/start-all.sh
```

Make sure Hadoop is running successfully. First use the `jps` command to determine which Java processes are running:

```
bob@scapevm:~$ jps
```

You should see the following processes: (ignore the process IDs)

```
2474 SecondaryNameNode
2933 Jps
2113 NameNode
2548 JobTracker
2724 TaskTracker
2293 DataNode
```

If there are any missing processes, you will need to review logs. By default, logs are located under:

```
/usr/local/hadoop/logs
```

Look through the log files for the Namenode and the Jobtracker in order to ensure that HDFS and MapReduce have been successfully started. In particular, look for Exceptions that might have occurred during start-up.

---

<sup>14</sup> <https://www.virtualbox.org/>



### Run your first Job

We will be running the word count example using an arbitrary text file as input. For the wordcount example you will need a plain text file large enough to be interesting. For this purpose, you can download plain text books from Project Gutenberg to the local file system (an example book can also be found in the Downloads folder).

Hadoop creates a home directory for every user on its distributed file system (HDFS). This directory is typically used to store (larger amounts of) input data to a MapReduce job. Access permission can be set similar to Unix file systems (see also [http://hadoop.apache.org/docs/r1.2.1/file\\_system\\_shell.html](http://hadoop.apache.org/docs/r1.2.1/file_system_shell.html)).

Now let's copy the books onto the HDFS file system by using the Hadoop dfs command. In the command String below, /home/bob/sampleBook.txt must be replaced with the path to your input file; /user/bob/sampleBook.txt specifies the path on HDFS.

```
bob@scapevm:~$ hadoop dfs -copyFromLocal /home/bob/sampleBook.txt
/user/bob/sampleBook.txt
```

To display the contents of your home directory on HDFS use the following command:

```
bob@scapevm:~$ hadoop dfs -ls /user/bob
```

To display the content of a file on HDFS use this command:

```
bob@scapevm:~$ hadoop dfs -cat /user/bob/sampleBook.txt
```

After copying the book(s) to HDFS, execute the wordcount example:

```
bob@scapevm:~$ hadoop jar /usr/local/hadoop/hadoop-examples-1.2.1.jar
wordcount /user/bob/books /user/bob/books-output
```

You will see the job running now; the results from the run should be in /user/bob/books-output in our case. To download the results file execute:

```
bob@scapevm:~$ hadoop dfs -copyToLocal /user/bob/books-output/part-r-00000
output.txt
```

### Monitoring HDFS/Jobs

The Hadoop management web sites allow you to monitor jobs, check log files, browse the file system, etc. The web interfaces to the JobTracker and the NameNode provide very helpful information about the status of the MapReduce and HDFS systems running on your machine.

The JobTracker allows you to see information regarding the cluster, running map and reduce tasks, node status, running and completed jobs. The monitoring site can be found at <http://localhost:50030/jobtracker.jsp>

The NameNode allows you to browse the HDFS system, view nodes, look at space consumption, and check logs. The DFSHealth site can be found at <http://localhost:50070/dfshealth.jsp>

### **Hadoop Command Reference**

The File System Shell Guide provides a full reference of HDFS shell commands:

`http://hadoop.apache.org/docs/r1.2.1/file\_system\_shell.html`

The Commands Guide provides a full list of Hadoop commands required to execute and monitor the system as well as individual jobs:

`http://hadoop.apache.org/docs/r1.2.1/commands\_manual.html`

### **Using the Guidelines**

Using the Hadoop guidelines, you can find out many details of your Hadoop cluster. Try to accomplish the following tasks:

- What is the configured capacity of your HDFS, how many data nodes are connected, and how many blocks have corrupted replicas?
- Execute the MapReduce Grep Example program against the previously used input files. The Grep program is documented here: <http://wiki.apache.org/hadoop/Grep>
- Start a Hadoop job from the command-line and, in another terminal window, monitor the default job queue and obtain the job identifier of your job.

## **Appendix B Instructions for Tiff2JP2-Example**

### **Integrating Taverna with Hadoop**

The goal of this example is to modify an existing preservation workflow so that it can be executed on a scalable environment. In order to fulfil this task we will make use of Taverna, Hadoop, and a MapReduce application for wrapping command-line tools.

### **Performing the task from the command-line**

In the following, we make use of a workflow that automatically converts TIFF images to JP2 (JPEG 2000) using OpenJPEG, followed by a format validation which is done using Jpylyzer. Open a terminal window and perform this task on the command-line against a single test file. A script `tiff2jp2.sh` that applies this task against multiple image files can be found in the folder `Tiff2JP2-Example`. Use the script to experiment with sets of sample images.

### **Implementing the Taverna workflow**

The task from the previous exercise can be easily implemented, visualized, and run as a Taverna workflow. The `Tiff2JP2-Example` directory contains a workflow called `tiff2jp2.t2flow` which implements this scenario. The workflow takes a reference to a directory as input and executes the tool chain for every TIFF file contained in that folder. Open the workflow in Taverna and investigate how the Taverna's tool invocation mechanism is used to execute the preservation tools. Also examine the remaining activities used to select the files and generate the proper command-line statement. Finally, run the workflow against a test data set and examine the results.



### Using the SCAPE MapReduce Tool Wrapper (ToMaR)

In this section we will use a MapReduce application to execute OpenJpeg against a set of images on Hadoop. The SCAPE MapReduce tool wrapper<sup>15</sup> (ToMaR) is used to execute (preservation) command-line tools on a Hadoop cluster. As Hadoop input data resides on the HDFS file system which is typically not supported by a conventional command-line tool, the wrapper is required to handle the communication between the Hadoop runtime environment and the tools to be executed. You can find the toolwrapper within the Tiff2JP2-Example folder as a Java archive (.jar file). A set of tool specification documents are provided within the toolspecs folder. These files describe different command-line patterns telling the tool wrapper how the user wants a particular tool being invoked (e.g. using various parameters). Take a look at the tool specification for OpenJpeg; we will make use of in the subsequent steps.

Now copy the entire toolspecs folder to your home directory on HDFS using the put command (`hadoop dfs -put toolspecs`). Verify that the toolspecs are available on HDFS using `-ls` and `-cat` commands. The toolwrapper also requires an input file that specifies the parameters for the tool invocations, i.e. the location of the input files and the output files. This file will be generated within the Taverna workflow. However, if the toolwrapper is used from the command-line, the input file has to be generated manually.

Use the script `gen-input.sh` to create an input file for the TIFF files you would like to migrate to JP2. For example type `./gen-input.sh files/*.tiff` to generate an input file for all .tiff-images in the files folder. Finally, we have to make the input files available on the distributed file system. Copy the folder with the input files you specified in the previous step to your home directory on HDFS. Make sure that the references used in the file `input-files.txt` correspond with location of your input files on HDFS.

Now, we are ready to start the tool wrapper. Use the following command to start the MapReduce application: `hadoop jar toolwrapper.jar -i input-files.txt -r toolspecs`. (The last parameter sets the location of the toolspecs on HDFS). Monitor the execution of your application using the web interfaces and log-files. The JP2-files will be put into the input directory. Output written by MapReduce will be put into a folder called out by default.

### Using Taverna to orchestrate Hadoop

In the following we will use a Taverna workflow that implements the Tiff2JP2 workflow by using Hadoop as the execution environment. The workflow `tiff2jp2_hadoop.t2flow` is available in the Tiff2JP2-Example directory and can be loaded into Taverna workbench just like any other workflow. Take a look at the tool invocation activities and compare them with the previously executed workflow, which is using local tool invocations. Also, note the Location tab of the tool configuration dialog. This is presently set to `"default local"`. While this configuration is fine for the virtual machine setup we are using, it would be required to add a new location to access a remote cluster. Use the "Manage locations" button to find out how remote locations can be added. The password of Taverna's Credential Manager is: `master`. Let's start the workflow on the (pseudo) Hadoop cluster that is running on your local virtual machine! Ensure the folder containing a set of Tiff images to process in this workflow is ready on your HDFS home directory. The workflow exposes two input ports: (1) for `hadoopjobjar` use "Set file location" and point Taverna to the `toolwrapper.jar`; (2) for `inputdir` use "Set value" and provide the path to your input files on HDFS, e.g. `/user/bob/myTiffs`.

---

<sup>15</sup> <https://github.com/openplanets/tomar>



Before you run the workflow, be sure to have the MapReduce Administration Interface open in a browser window. Finally, execute the workflow and monitor the execution of the different MapReduce jobs as well as the created results.

### **Workflow + Image Processing**

For a version of this workflow that is extended by image processing activities for quality assurance, take a look at the directory “Tiff2JP2withImgProcessing-Example”. The folder contains a more complex version of the Tiff to JP2 migration workflow, which for example also makes use of the SCAPE Matchbox tool to visually compare the source and target files. This workflow can also be run on your virtual machine as described above.

## Appendix C Instructions for Cluster Setup

### Desktop Setup

The virtual image-based release is configured to run Hadoop in pseudo-distributed mode per default. This allows a user to start a Hadoop instance on a single device (even if it has only a single processor core) by running various Hadoop daemons based on threads. While this setup is very helpful for experimenting with Hadoop or for testing and debugging MapReduce applications, it is not suitable for scalable processing as it only mimics parallel execution. It is however only a matter of configuration to use multiple instances of the virtual machine image to create a cluster which is equipped with a set of pre-installed preservation tools on its nodes. The following provides a brief overview on deploying the VM release in a parallel environment. For more detailed instructions on configuring Hadoop, the reader is referred to the corresponding Hadoop release documentation<sup>16</sup>.

### Using Multiple Cores

Hadoop can be configured to efficiently execute multiple map and reduce on a multi-core node, independently if the machine is running standalone (in pseudo-distributed mode) or in a (real distributed) cluster setup. For enabling multiple map and reduce tasks the file `mapred-site.xml` available in Hadoop's "conf" directory must be adapted (and/or tuned with respect to the processor hardware):

- Add the properties `mapreduce.tasktracker.map.tasks.maximum` and `mapreduce.tasktracker.reduce.tasks.maximum` to control the number of map and reduce tasks per node.

- Add the properties `mapreduce.job.maps` and `mapreduce.job.reduces` to control the default number of map and reduce tasks per job.

### Cluster Setup

Using the VM release on multiple computers allows you to create a distributed (scalable) Hadoop cluster which has the preservation tools described by the previously discussed SCAPE tool specification documents already pre-installed on the nodes, so that they can be executed in parallel using ToMaR and produce results that are readily available on the Hadoop infrastructure, for example through the HDFS file system.

In the following, we assume that all cluster nodes running the VM release - either as virtual machine or natively – are connected through a local network<sup>17</sup>. Configure a static IP address on the cluster nodes by adapting the file `/etc/network/interfaces`. Use the file to configure IP address, netmask, and gateway according to your environment. Ensure that a DNS server is configured by checking the file `/etc/resolve.conf`. The next step before the cluster can be started is to configure the master node and the slave nodes. In the following we provide an outline of the required configuration.

---

<sup>16</sup> [http://hadoop.apache.org/docs/r1.2.1/cluster\\_setup.html](http://hadoop.apache.org/docs/r1.2.1/cluster_setup.html)

<sup>17</sup> In order to setup your own private network for the cluster it will be required to set up a router on the cluster front-end computer. Please consult the Debian and Ubuntu documentation on setting up a NAT router accordingly.



- Here, we assume that the host name of the cluster front-end node is "master" and that the worker nodes have the names "node0" – "nodeX". Add the nodes to the /etc/hosts file on the nodes or configure a local DNS server to accomplish name resolution.
- On the master node, edit /conf/master to add the domain name of the master node and edit /conf/slaves to add domain names of the slave nodes.
- On all nodes, edit /conf/core-site.xml and replace localhost by the master domain name for the property fs.default.name, and similarly edit /conf/mapred-site.xml and update the property mapred.job.tracker.
- From the master node, format the HDFS file system using the command "hadoop namenode -format" and subsequently start the cluster using the script "start-all.sh".
- Use the "jps" command to verify that on the master NameNode, SecondaryNameNode, and JobTracker (optionally also Tasktracker and DataNode), and on the slaves TaskTracker and DataNode daemons are running.

Also consult the previously mentioned JobTracker status page (<http://master:50030/jobtracker.jsp>) and the NameNode status page (<http://master:50070/dfshealth.jsp>) on the master to see detailed information on the cluster's MapReduce and HDFS sub-systems.