D2.7 System Integration and Validation Test Plan

WP2 – EO Platform

Deliverable Lead: ATOS FR

Dissemination Level: Public

Deliverable due date: 31/12/2016

Actual submission date: 22/12/2016

Version 1.2
### Document Control Page

<table>
<thead>
<tr>
<th>Title</th>
<th>D2.7 System Integration and Validation Test Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creator</td>
<td>Anne Haugommard (ATOS FR)</td>
</tr>
<tr>
<td>Description</td>
<td>This document describes the integration system and validation test plan of the platform.</td>
</tr>
<tr>
<td>Publisher</td>
<td>EO4wildlife Consortium</td>
</tr>
<tr>
<td>Contributors</td>
<td>Anne Haugommard (ATOS FR)</td>
</tr>
<tr>
<td></td>
<td>Fanny Martin (ATOS FR)</td>
</tr>
<tr>
<td>Creation date</td>
<td>18/10/2016</td>
</tr>
<tr>
<td>Type</td>
<td>Text</td>
</tr>
<tr>
<td>Language</td>
<td>en-GB</td>
</tr>
<tr>
<td>Rights</td>
<td>copyright “EO4wildlife Consortium”</td>
</tr>
<tr>
<td>Audience</td>
<td>☑ Public</td>
</tr>
<tr>
<td></td>
<td>☐ Confidential</td>
</tr>
<tr>
<td></td>
<td>☐ Classified</td>
</tr>
<tr>
<td>Status</td>
<td>☐ In Progress</td>
</tr>
<tr>
<td></td>
<td>☐ For Review</td>
</tr>
<tr>
<td></td>
<td>☐ For Approval</td>
</tr>
<tr>
<td></td>
<td>☑ Approved</td>
</tr>
</tbody>
</table>

### Disclaimer

This deliverable is subject to final acceptance by the European Commission.

The results of this deliverable reflect only the author’s view and the Commission is not responsible for any use that may be made of the information it contains.

### Statement for open documents

(c) 2016 EO4wildlife Consortium

The EO4wildlife Consortium (http://eo4wildlife.eu) grants third parties the right to use and distribute all or parts of this document, provided that the EO4wildlife project and the document are properly referenced.
# Table of Contents

EO4wildlife Project Overview ................................................................................................. 5
Executive Summary .................................................................................................................. 6

1 System overview .................................................................................................................. 7
   1.1 System components ......................................................................................................... 7
       1.1.1 Distributed file system ......................................................................................... 8
       1.1.2 SparkInData services ......................................................................................... 9
       1.1.3 GeoServer .......................................................................................................... 9
       1.1.4 GeoNetwork ...................................................................................................... 10
       1.1.5 Processing services .......................................................................................... 10
       1.1.6 EO4wildlife dedicated webservices ..................................................................... 10
       1.1.7 Front-end ........................................................................................................... 10
   1.2 Component interactions ................................................................................................ 11

2 Integration and validation strategy ..................................................................................... 13
   2.1 Integration process ......................................................................................................... 13
       2.1.1 Roles and responsibilities .................................................................................... 13
       2.1.2 Integration and validation workflow ..................................................................... 13
       2.1.3 Anomaly report management ............................................................................. 15
   2.2 Integration strategy ........................................................................................................ 15
   2.3 Integration and validation test infrastructure ................................................................ 16
       2.3.1 Integration infrastructure .................................................................................... 18
       2.3.2 Validation infrastructure .................................................................................... 18
       2.3.3 Production infrastructure ................................................................................... 18
   2.4 Assumptions and limitations ......................................................................................... 19

3 Integration test cases .......................................................................................................... 20
   3.1 Components deployment ............................................................................................... 20
       3.1.1 EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA .................................... 20
       3.1.2 EO4WL_INT_TEST_002_DEPLOYMENT_CMEMS_CONNECTOR ............................ 20
       3.1.3 EO4WL_INT_TEST_003_DEPLOYMENTTRACK_LOC_WPS ............................... 21
       3.1.4 EO4WL_INT_TEST_004_DEPLOYMENT_BIRDLIFE_WPS ................................ 21
       3.1.5 EO4WL_INT_TEST_005_DEPLOYMENT_WEBSERVICES .................................. 22
       3.1.6 EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END ....................................... 22
   3.2 Webservices .................................................................................................................. 23
       3.2.1 EO4WL_INT_TEST_007_WEBSERVICES_FILE_MANAGER ................................ 23
       3.2.2 EO4WL_INT_TEST_008_WEBSERVICES_EXECUTE_PROCESS ............................ 23
       3.2.3 EO4WL_INT_TEST_009_WEBSERVICES_PROCESS_STATUS .............................. 23
   3.3 Front end ....................................................................................................................... 24
       3.3.1 EO4WL_INT_TEST_010_FRONTEND_GEONETWORK ....................................... 24
       3.3.2 EO4WL_INT_TEST_011_FRONTEND_FILEMANAGER ......................................... 24
       3.3.3 EO4WL_INT_TEST_012_FRONTEND_FILEMANAGER ......................................... 25
       3.3.4 EO4WL_INT_TEST_013_FRONTEND_GEOSERVER ........................................... 25
       3.3.5 EO4WL_INT_TEST_014_FRONTEND_GEOSERVER ........................................... 26
       3.3.6 EO4WL_INT_TEST_015_FRONTEND_EXECUTE_PROCESS .................................. 26
       3.3.7 EO4WL_INT_TEST_016_FRONTEND_PROCESS_STATUS ..................................... 27

4 Validation test cases ........................................................................................................... 28

5 Conclusion .......................................................................................................................... 29

References ............................................................................................................................... 30

Annex I - Script integration detailed on Birdlife WPS service .................................................. 31
List of Figures

Figure 1: EO4wildlife system components overview ................................................................. 7
Figure 2: Directories structure for EO4wildlife workspaces into distributed file system ............... 8
Figure 3: Sample content of a user workspace directory ............................................................ 8
Figure 4: Versions of the services available into GeoServer ....................................................... 10
Figure 5: Component interactions overview ............................................................................. 11
Figure 6: Overview of the integration and validation workflow .................................................. 14
Figure 7: Integration steps .......................................................................................................... 16
Figure 8: overview of the integration and validation environments ............................................ 17
Figure 9: Overview of the integration infrastructure .................................................................. 18

List of Tables

Table 1: Component interfaces overview .................................................................................... 12
EO4wildlife Project Overview

EO4wildlife main objective is to bring large number of multidisciplinary scientists such as biologists, ecologists and ornithologists around the world to collaborate closely together while using European Sentinel Copernicus Earth Observation more heavily and efficiently.

In order to reach such important objective, an open service platform and interoperable toolbox will be designed and developed. It will offer high level services that can be accessed by scientists to perform their respective research. The platform front end will be easy-to-use, access and offer dedicated services that will enable them process their geospatial environmental stimulations using Sentinel Earth Observation data that are intelligently combined with other observation sources.

Specifically, the EO4wildlife platform will enable the integration of Sentinel data, ARGOS archive databases and real time thematic databank portals, including Wildlifetracking.org, Seabirdtracking.org, and other Earth Observation and MetOcean databases; locally or remotely, and simultaneously.

EO4wildlife research specialises in the intelligent management big data, processing, advanced analytics and a Knowledge Base for wildlife migratory behaviour and trends forecast. The research will lead to the development of web-enabled open services using OGC standards for sensor observation and measurements and data processing of heterogeneous geospatial observation data and uncertainties.

EO4wildlife will design, implement and validate various scenarios based on real operational use case requirements in the field of wildlife migrations, habitats and behaviour. These include:

- Management tools for regulatory authorities to achieve real-time advanced decision-making on the protection of protect seabird species;
- Enhancing scientific knowledge of pelagic fish migrations routes, reproduction and feeding behaviours for better species management;
- Enable researchers better understand the movement behaviour of sea turtle populations; and
- Setting up tools to assist marine protected areas and management.

Abbreviations and Glossary

A common glossary of terms for all EO4wildlife deliverables, as well as a list of abbreviations, can be found in the public document “EO4wildlife Glossary” available at EO4wildlife.eu.
Executive Summary

The objective of this system integration and validation test plan is to describe the system components and interactions, the integration strategy and associated testing environments and the integration tests.

The system is described in section 1, with a description of the component structure and interactions between the different components.

The integration strategy is presented in section 2, with the description of the integration process in section 2.1 with the roles and responsibilities, the integration and validation workflow. The steps of the integration procedure are described in section 2.2 and a description of integration, validation and production environments is provided in section 2.3.

The integration tests are detailed in section 3 of the document.

An example describing the script integration steps from a R (BirdLife) script to a WPS processing service is provided as Annex at the end of the deliverable.
1System overview

This section presents the EO4wildlife system components and their interactions.

1.1 System components

The following schema presents the overview of the EO4wildlife system components.

![EO4wildlife system components overview](image)

**Figure 1**: EO4wildlife system components overview

EO4wildlife system components are the following:

- **EO4wildlife application**, which is composed of:
  - A distributed file system, where user spaces and common spaces are stored
  - SparkInData services, handling user authentication management, application deployment and data management service.
  - GeoServer which is in charge of WPS process launching
  - GeoNetwork which is in charge of the product catalogue
  - Dedicated EO4wildlife webservices
  - Front-end

- **Processing services** provided by partners that are made available as WPS processes
1.1.1 Distributed file system

The distributed file system used is provided by GlusterFS component of SparkInData platform. Content is organized into directories, and EO4wildlife dedicated directory is located under /data/exchange/eo4wildlife.

The sub-directories are structured in the following way:

![Directories structure](image)

**Figure 2**: Directories structure for EO4wildlife workspaces into distributed file system

1.1.1.1 Common data directory

A “common_data” directory contains a set of data not available through connectors, and made available to all EO4wildlife users:

- These common data are organized into subfolders according to the kind of data provided.
- Only platform administrator has write access to this directory, other users have read access.
- Any request to add data to this directory must be submitted directly to platform administrators.

1.1.1.2 User workspace directories

Each user of EO4wildlife platform has a user workspace directory named after the user login. The subfolders are organized in the following way:

![Sample content of a user workspace directory](image)

**Figure 3**: Sample content of a user workspace directory

A “process_dir” directory is present and for each processing service execution, a subdirectory named after the identifier of the process execution (exec001 in the previous example) is created.

At the end of process execution, it contains an “out” directory containing the output files of the processing service.
The other subdirectories are more technical and used during the process execution:

- **config**: configuration files needed for the process, either given as process input or generated during the process if it depends on some other inputs (not used for Birdlife and Track&Loc use cases)
- **in**: symbolic links to the input data of the processing service, or copies of the data if inputs are external URLs
- **tmp**: folder used during the process execution for intermediate files. This folder is empty at the end of a successful process.

### 1.1.2 SparkInData services

The SparkInData services are described into system architecture document [3]. The main services used by EO4wildlife are the following:

- Data management service, in charge of handling the metadata catalogue (GeoNetwork, see section 1.1.4), the ingestion of the associated data, and the data lifecycle;
- Watching service, in charge of feeding the metadata catalogue;
- Service manager, in charge of the deployment and monitoring of new applications.

EO4wildlife platform is based on v1.0 of SparkInData platform.

### 1.1.3 GeoServer

GeoServer is an open source component for sharing of geospatial data, and execution of processing services.

In the context of EO4wildlife application, the following functionalities of GeoServer are used:

- Definition of new processing services
- Execution of processing services through WPS standard through a WPS extension of GeoServer
- Web Feature Service (WFS) support
- Web Map Service (WMS) support
- Web Coverage Service (WCS) support
- Graphical user interface
- REST API

GeoServer is available from: [http://geoserver.org](http://geoserver.org).

Version 2.8.1 is deployed into the platform.

The following versions of the services are available into GeoServer. Detailed operations available for each service are described into [1].
1.1.4 GeoNetwork

GeoNetwork is used for metadata catalogue handling, as described in System architecture document [3]. GeoNetwork is available from [http://geonetwork-opensource.org](http://geonetwork-opensource.org).

In the context of EO4wildlife application, the following functionalities of GeoNetwork are used:

- Metadata catalogue management
- CSW endpoint

Version 3.0.4 is deployed into the platform.

1.1.5 Processing services

Scripts for processing services are provided by partners and integrated into the platform through the procedure described in Annex I - Script integration.

1.1.6 EO4wildlife dedicated webservices

SparkInData offers a first set of generic services, and EO4wildlife has a need for additional specific services on top of SparkInData services and components.

The following dedicated webservices are developed for EO4wildlife:

- **FileManagerService**, which handles the interactions with the distributed file system and provides the following functionalities:
  - File upload
  - File list
  - File download
- **ExecuteProcessService**, which handles the execution of WPS services;
- **ExecutionStatusService**, which handles the monitoring of the WPS service executions.

1.1.7 Front-end

The Front-end component is made available through a web application deployed into an Apache server.
The following technologies are used inside the Front end component:

- **Angular JS**
  - Angular JS is an open source JavaScript framework developed by Google for web application development. See [https://angularjs.org](https://angularjs.org)

- **IFrames**
  - `<iframe>` is an HTML tag used to encapsulate parts of external websites. In EO4wildlife context, it will be used to encapsulate a part of GeoNetwork user interface and GeoServer user interface into EO4wildlife front-end.

- **Angular File Manager**, which is a JavaScript library for File Management Explorer available at: [https://github.com/joni2back/angular-filemanager](https://github.com/joni2back/angular-filemanager)

- **OpenLayers**
  - OpenLayers is an open source JavaScript library used to query the GeoServer WMS and renders the map visualization to the web browser.

### 1.2 Component interactions

The following schema presents the interactions between the system components.

![Component interactions overview](image)

**Figure 5**: Component interactions overview

EO4wildlife is deployed as a SparkInData service into SparkInData platform deployed on Helix Nebula cloud environment.
EO4wildlife application contains an Apache HTTP Server, hosting both Front-end web application and EO4wildlife back-end web services.

Front-end application is accessible from a web browser on a user personal computer and interacts with:

- EO4wildlife dedicated web services;
- GeoNetwork component, embedding parts of GeoNetwork user interface;
- GeoServer component, embedding parts of GeoServer user interface.

EO4wildlife web services:

- Are built upon SparkInData core services as described in architecture document [3];
- Manage the access to the EO4wildlife directories on the distributed file system (user workspaces and common workspace);
- Manage processing services through calls to GeoServer services.

Each of these interactions will be tested during integration tests.

The interfaces presented in Figure 5 are described in the following table:

<table>
<thead>
<tr>
<th>#</th>
<th>Components involved</th>
<th>Protocol / Standard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User browser</td>
<td>Front end</td>
<td>HTTP</td>
</tr>
<tr>
<td>2</td>
<td>Front end</td>
<td>Webservices</td>
<td>HTTP/REST</td>
</tr>
<tr>
<td>3</td>
<td>Front-end</td>
<td>GeoServer</td>
<td>HTTP/WPS</td>
</tr>
<tr>
<td></td>
<td>GeoServer</td>
<td>HTTP/WFS</td>
<td>Front-end has an access point to GeoServer managed resources using OGC WFS standard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HTTP/WMS</td>
<td>Front-end has an access point to GeoServer managed resources using OGC WMS standard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HTTP/WCS</td>
<td>Front-end has an access point to GeoServer managed resources using OGC WCS standard</td>
</tr>
<tr>
<td>4</td>
<td>Front-end</td>
<td>GeoNetwork</td>
<td>HTTP / CSW</td>
</tr>
<tr>
<td>5</td>
<td>Webservices</td>
<td>DFS</td>
<td>HTTP</td>
</tr>
<tr>
<td>6</td>
<td>Webservices</td>
<td>GeoServer</td>
<td>HTTP/WPS</td>
</tr>
<tr>
<td>7</td>
<td>Webservices</td>
<td>SparkInData core services</td>
<td>HTTP/REST</td>
</tr>
</tbody>
</table>

*Table 1: Component interfaces overview*
2 Integration and validation strategy

The integration and validation process, strategy and infrastructure are described in the following sections.

2.1 Integration process

The integration process defines roles and a workflow for integration.

2.1.1 Roles and responsibilities

The following roles have been identified in the integration and validation process:

- **Development teams**
  Several development teams are working in parallel, with different purposes. Three kinds of development teams have been identified:
  
  - **Service providers**, who are in charge of providing the processing services described in section 1.1.5
  - **Platform provider**, who is in charge of providing EO4wildlife dedicated platform services described in section 1.1.6
  - **Front end provider**, who is in charge of providing the front-end component described in section 1.1.7

- **Integration team**, who is in charge of the following tasks:
  
  - Defining integration infrastructure as described in section 2.3.1
  - Defining integration test cases as described in section 3
  - Integrating all components into a platform based upon SparkInData platform provided by SparkInData team as described in section 2.1.2.2
  - Running integration tests into integration infrastructure and reporting test results and associated issues as described in section 2.1.3.

- **Validation team**, who is in charge of the following tasks:
  
  - Defining validation tests as described in Validation plan [2]
  - Running validation tests and reporting test results and associated issues as described in section 2.1.3.

- **User**, who is a final user of the deployed application.

2.1.2 Integration and validation workflow

The following schema summarizes the integration and validation workflow.
2.1.2.1 Development phase

Each development team works on a different development environment according to its own needs. Service providers, front-end provider, platform provider, and data connector providers do not need the same tooling. These development environments are local for each partner. Each development team is in charge of its own components unitary tests inside this environment.

2.1.2.1.1 Service development

As described in section 1.1.5, service providers develop the scripts and deliver them to integration team. The delivery of a processing service to integration team consists of the following content:

- The source code of the script (R, Python...)
- A detailed description of
  - The execution environment,
  - The inputs and outputs of the scripts,
  - The execution command

2.1.2.1.2 EO4wildlife platform development

As described in sections 1.1.2 and 1.1.6, SparkInData platform is delivered by SparkInData project team and EO4wildlife dedicated webservices are developed on top of SparkInData services.

2.1.2.1.3 Front-end development

As described in section 1.1.7 the Front-end is developed on top of EO4wildlife web services.

Web services REST API specifications are exchanged using OpenAPI standard (see https://www.openapis.org), and particularly using Swagger format (http://swagger.io).
Front-end sources are delivered to integration team.

2.1.2.2 Integration phase

The integration phase aims at testing the technical integration and deployment of components into a common infrastructure similar to the production environment. Integration tests are performed by the integration team in order to ensure a correct deployment of the components and valid interactions between the components.

The inputs of the integration phase are the following:

- Common data identified from processing services expected inputs
- SparkInData platform
- Sources or the processing services scripts
- Sources of the dedicated EO4wildlife webs services
- Sources of the front-end web application

As soon as these inputs are available, the integration phase is launched: it consists in applying the integration strategy described section 2.2 and for the integration team to run the tests described in section 3.

The integration environment is described in section 2.3.1.

2.1.2.3 Functional validation phase

The functional validation phase is performed once the integration phase is successful, which means that all integration tests have been run without any remaining blocking or major issue.

The objective of the functional validation phase is for the validation team to test the functional behaviour of the platform using tests described into dedicated deliverable Evaluation and validation Plan [2].

The functional validation environment is described in section 2.3.2.

2.1.2.4 Deployment phase

Once the functional validation is successful, the platform is deployed into Product environment, accessible to external users into a SparkInData infrastructure.

2.1.3 Anomaly report management

During integration phase, all issues are reported through an internal ATOS FR managed tool. Test reports are transmitted to EO4wildlife partners during integration phase.

2.2 Integration strategy

The figure below describes the steps for the integration of the EO4wildlife platform.
The first step is the deployment of the SparkInData platform.

On top of this infrastructure environment,

- Processing services can be integrated into GeoServer component as WPS processing services, performing the following steps (as described in details into the example in Annex I – Script integration):
  - First the script provided as source code by service providers is encapsulated into a Docker container, with inputs, outputs and launching commands clearly identified,
  - Then this Docker is registered into platform DockerRegistry in order to be accessible for launching.
  - The processing service is integrated into GeoServer and JobScheduler components as a GeoServerProcess calling the Docker files with correct mapping of input and output directories on the distributed file system.

- Data connectors are integrated into the platform on top of DataManagementServer and GeoNetwork catalogue

- Dedicated EO4wildlife webservices can be integrated.

Front-end application can be integrated with EO4wildlife webservices into a Docker container and then as a SparkInData service deployed on the SparkInData marketplace.

After an initialization of users’ accounts and common data directory, the EO4wildlife application is integrated.

2.3 Integration and validation test infrastructure

The following schema presents the different necessary infrastructures.
According to the integration and validation workflow described section 2.1.2, several environments are necessary:

- One dedicated development environment for each development team (handled by the team itself). These environments are not hosted on a cloud infrastructure.
- An integration environment available for integration team during integration phase.
- A functional validation environment available for validation team during validation phase.
- A production environment available for EO4wildlife users after a successful validation process.

**Figure 8:** overview of the integration and validation environments
2.3.1 Integration infrastructure

The material infrastructure for integration phase consists in a set of virtual machines deployed into Helix Nebula cloud, hosting SparkInData platform components:

- GlusterFS distributed file system (replicated) and MongoDB database (replicated)
- Kubernetes in which are deployed the following items:
  - GeoNetwork instance
  - GeoServer instance
  - SparkInData services (Service Manager, Data management service, Ingestion service, Watching service)
  - EO4wildlife web services
  - EO4wildlife front-end

This environment is deployed on Helix Nebula cloud and available to integration team.

2.3.2 Validation infrastructure

The validation environment is a duplicate of the integration environment in terms of material infrastructure.

In terms of access, this environment is deployed on Helix Nebula cloud and made accessible to EO4wildlife partners, in order for validation team to perform the tests.

2.3.3 Production infrastructure

The validation environment is similar to the validation environment in terms of infrastructure.

In terms of access, this environment is deployed on Helix Nebula cloud and made accessible to users. EO4wildlife partners have user accounts on this environment.
2.4 Assumptions and limitations

The content of this document corresponds to EO4wildlife platform v2 plans.
3 Integration test cases

Integration tests are organized into the following categories:

- SparkInData deployment
- EO4wildlife
- Processing services, in order to test the availability of the processing services into GeoServer
- Webservices
- Front-end
- Data management.

The integration tests identified are described below.

3.1 Components deployment

3.1.1 EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA

- **Test objective**: Ensures the availability of the different SparkInData components deployed
- **Prerequisites**:
  - SparkInData platform is deployed
  - A test user has been created
- **Test steps**:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access platform URL through web browser</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>Connect as test user</td>
<td>Test user is connected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SparkInData market place page is accessible</td>
</tr>
<tr>
<td>3</td>
<td>Access GeoNetwork deployment through direct URL based on ip address</td>
<td>GeoNetwork is accessible</td>
</tr>
<tr>
<td>4</td>
<td>Access GeoServer deployment through direct URL based on ip address</td>
<td>GeoServer is accessible</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: N/A

3.1.2 EO4WL_INT_TEST_002_DEPLOYMENT_CMEMS_CONNECTOR

- **Test objective**: Ensures the deployment of CMEMS connector into SparkInData DataManagementService
- **Prerequisites**:
  - SparkInData platform is deployed
  - A test user has been created
  - EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA successful
- **Test steps**:
## Description

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access platform URL through web browser</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>Connect as test user</td>
<td>SparkInData market place page is accessible</td>
</tr>
<tr>
<td>3</td>
<td>Access GeoServer deployment through direct URL based on ip address</td>
<td>GeoNetwork is accessible</td>
</tr>
<tr>
<td>4</td>
<td>Perform a CSW search on a particular CMEMS product, for example:</td>
<td>Product metadata is accessible in GeoNetwork catalogue</td>
</tr>
<tr>
<td></td>
<td>Global Ocean, Ocean Colour Chlorophyll (OCEANCOLOUR_GLO_CHL_L3_NRT_OBSERVATIONS_009_032)</td>
<td></td>
</tr>
</tbody>
</table>

**Interfaces under tests:** CMEMS Connector

### 3.1.3 EO4WL_INT_TEST_003_DEPLOYMENT_TRACK_LOC_WPS

- **Test objective:** Ensures that Track&Loc is deployed as a WPS process into GeoServer
- **Prerequisites:**
  - SparkInData platform is deployed
  - A test user has been created
  - EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA successful
- **Test steps:**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access platform URL through web browser</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>Connect as test user</td>
<td>SparkInData market place page is accessible</td>
</tr>
<tr>
<td>3</td>
<td>Access GeoServer deployment through direct URL based on ip address</td>
<td>GeoServer is accessible</td>
</tr>
<tr>
<td>4</td>
<td>Select WPS process list page</td>
<td>Track&amp;Loc processing service is accessible from the list of processes</td>
</tr>
<tr>
<td>5</td>
<td>Select Track&amp;Loc process from the list</td>
<td>Check that inputs and outputs displayed correspond to process expected inputs and outputs</td>
</tr>
</tbody>
</table>

**Interfaces under tests:** WPS Process integration

### 3.1.4 EO4WL_INT_TEST_004_DEPLOYMENT_BIRDLIFE_WPS

- **Test objective:** Ensures that Birdlife script is deployed as a WPS process into GeoServer
- **Prerequisites:**
  - SparkInData platform is deployed
  - A test user has been created
  - EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA successful
- **Test steps:**
D2.7 System Integration and Validation Test Plan

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access platform URL through web browser</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>Connect as test user</td>
<td>SparkInData market place page is accessible</td>
</tr>
<tr>
<td>3</td>
<td>Access GeoServer deployment through direct URL based on ip address</td>
<td>GeoServer is accessible</td>
</tr>
<tr>
<td>4</td>
<td>Select WPS process list page</td>
<td>Birdlife scenario processing service is accessible from the list of processes</td>
</tr>
<tr>
<td>5</td>
<td>Select the process from the list</td>
<td>Check that inputs and outputs displayed correspond to process expected inputs and outputs</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: WPS Process integration

### 3.1.5 EO4WL_INT_TEST_005_DEPLOYMENT_WEBSERVICES

- **Test objective**: Check that webservises are deployed and accessible

- **Prerequisites**:
  - SparkInData platform is deployed
  - A test user has been created
  - EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA successful
  - EO4wildlife webservises are deployed

- **Test steps**:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access EO4wildlife FileManager webservice through HTTP REST request</td>
<td>Webservice is accessible and returns a response.</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: Webservises deployment

### 3.1.6 EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END

- **Test objective**: Check the deployment of Front-end component and accessibility

- **Prerequisites**:
  - SparkInData platform is deployed
  - A test user has been created
  - EO4WL_INT_TEST_001_DEPLOYMENT_SPARKINDATA successful
  - EO4wildlife application (webservises and Front-end) is deployed as SEEED application

- **Test steps**:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access platform URL through web browser</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>Connect as test user</td>
<td>SparkInData market place page is accessible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EO4wildlife application is visible on MarketPlace applications</td>
</tr>
<tr>
<td>3</td>
<td>Select EO4wildlife application</td>
<td>Front-end home page is displayed</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: User / Front-end
3.2 Webservices

3.2.1 EO4WL_INT_TEST_007_WEBSERVICES_FILE_MANAGER

- **Test objective:** List files available on the common workspace (without the Front-end access)

- **Prerequisites:**
  - EO4WL_INT_TEST_005_DEPLOYMENT_WEBSERVICES
  - Initial data are available in the common workspace

- **Test steps:**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
</table>
  | 1 | Through HTTP/REST access, call the webservice `FileManagerServer` with
    - “List files” request
    - on common workspace directory | Request returns the file list |

- **Interfaces under tests:** Webservices / DFS

3.2.2 EO4WL_INT_TEST_008_WEBSERVICES_EXECUTE_PROCESS

- **Test objective:** Execute a WPS process deployed on GeoServer through Webservice (without the Front-end access)

- **Prerequisites:**
  - EO4WL_INT_TEST_005_DEPLOYMENT_WEBSERVICES
  - A test process is deployed into GeoServer and associated testing input data is available into the workspaces

- **Test steps:**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
</table>
  | 1 | Through HTTP/REST access, call the webservice `ExecuteProcessService` with
    - a manually set path on the DFS where the inputs are available
    - a WPS process description file filled manually | Request returns no error |
  | 2 | Through a direct connection on GeoServer instance, check the execution status of the process | Process is successful |
  | 3 | Check at the expected output path location on file system | Result files are available and correspond to the expected output file format |

- **Interfaces under tests:** Webservices / GeoServer

3.2.3 EO4WL_INT_TEST_009_WEBSERVICES_PROCESS_STATUS

- **Test objective:** Monitor status of WPS process executions through Webservice without Front-end

- **Prerequisites:**
  - EO4WL_INT_TEST_005_DEPLOYMENT_WEBSERVICES

- **Test steps:**
# Description | Expected result
---|---
1 | Through HTTP/REST requests, call web service `ExecuteProcessService` to launch 3 process executions:  
- One with invalid input that make execution fail  
- One successfully executed  
- One in progress | Status of the 3 processes are displayed:  
- Successful process  
- Failed process  
- In progress process
2 | Connect to GeoServer UI status page | Status of the 3 processes are displayed:  
- Successful process  
- Failed process  
- In progress process
3 | Through HTTP/REST request, call web service `ExecutionStatusService` to get the status of the processes | Status of the 3 processes are returned as response:  
- Successful process  
- Failed process  
- In progress process

- **Interfaces under tests:** Webservice `ExecutionStatusService` / GeoServer

## 3.3 Front end

### 3.3.1 EO4WL_INT_TEST_010_FRONTEND_GEONETWORK

- **Test objective:** GeoNetwork catalogue is accessible from Front-end
- **Prerequisites:**  
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END
- **Test steps:**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test user accesses EO4wildlife front-end</td>
<td>Home page is displayed</td>
</tr>
<tr>
<td>2</td>
<td>User selects DiscoverProduct page</td>
<td>GeoNetwork UI is accessible from EO4wildlife front-end</td>
</tr>
</tbody>
</table>

- **Interfaces under tests:** Front-end/GeoNetwork

### 3.3.2 EO4WL_INT_TEST_011_FRONTEND_FILEMANAGER

- **Test objective:** Check the interface between Front-end and webservices on the `FileManagerService` access
- **Prerequisites:**  
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END  
  - EO4WL_INT_TEST_007_WEBSERVICES_FILE_MANAGER
- **Test steps:**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Description</td>
<td>Expected result</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Test user connects to platform from URL</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>User navigates to marketplace</td>
<td>EO4wildlife application is available</td>
</tr>
<tr>
<td>3</td>
<td>User selects EO4wildlife application</td>
<td>EO4wildlife home page is displayed</td>
</tr>
<tr>
<td>4</td>
<td>User selects “browse workspace” page</td>
<td>User workspace and common workspace contents are displayed and can be explored</td>
</tr>
<tr>
<td>5</td>
<td>Check that a specific file is available from common workspace</td>
<td>File is available</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: Front-end/Webservices

### 3.3.3 EO4WL_INT_TEST_012_FRONTEND_FILEMANAGER

- **Test objective**: Test the availability of the following functionalities through webservice FileManagerService:
  - File upload
  - File download
  - File browse

- **Prerequisites**:
  - A test user is created with login “test_user”
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END
  - EO4WL_INT_TEST_011_FRONTEND_FILEMANAGER

- **Test steps**:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test user connects to platform from URL</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>User navigates to marketplace</td>
<td>EO4wildlife application is available</td>
</tr>
<tr>
<td>3</td>
<td>User selects EO4wildlife application</td>
<td>EO4wildlife home page is displayed</td>
</tr>
<tr>
<td>4</td>
<td>User selects “browse workspace” page</td>
<td>User workspace and common workspace contents are displayed and can be explored</td>
</tr>
<tr>
<td>5</td>
<td>Test user select directory “input_dir” and uploads a file to its private workspace</td>
<td>File is available at the following path /data/exchange/eo4wildlife/test_user/input_dir on the distributed file system</td>
</tr>
<tr>
<td>6</td>
<td>Test user browses its workspace</td>
<td>File is listed in the input_dir directory</td>
</tr>
<tr>
<td>7</td>
<td>Test user selects the file and selects the download menu</td>
<td>File is downloaded</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: Front-end / webservices

### 3.3.4 EO4WL_INT_TEST_013_FRONTEND_GEOSERVER

- **Test objective**: Check that Front-end gets the list of available processes from GeoServer

- **Prerequisites**:
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END

- **Test steps**:
D2.7 System Integration and Validation Test Plan

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test user connects to platform from URL</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>User navigates to marketplace</td>
<td>EO4wildlife application is available</td>
</tr>
<tr>
<td>3</td>
<td>User selects EO4wildlife application</td>
<td>EO4wildlife home page is displayed</td>
</tr>
<tr>
<td>4</td>
<td>User selects Process execution page</td>
<td>The list of processes accessible on GeoServer is displayed</td>
</tr>
<tr>
<td>5</td>
<td>Connect to GeoServer UI and display process list</td>
<td>The list is identical</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: Front-end / GeoServer WPS GetCapabilities

3.3.5 EO4WL_INT_TEST_014_FRONTEEND_GEOSERVER

- **Test objective**: Check that Front-end gets the list of available process inputs description from GeoServer

- **Prerequisites**:
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END
  - EO4WL_INT_TEST_003_DEPLOYMENT_TRACK_LOC_WPS

- **Test steps**:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test user connects to platform from URL</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>User navigates to marketplace</td>
<td>EO4wildlife application is available</td>
</tr>
<tr>
<td>3</td>
<td>User selects EO4wildlife application</td>
<td>EO4wildlife home page is displayed</td>
</tr>
<tr>
<td>4</td>
<td>User selects Process execution page</td>
<td>The list of processes accessible on GeoServer is displayed</td>
</tr>
<tr>
<td>5</td>
<td>User selects Trac&amp;Loc process</td>
<td>The process input form is displayed</td>
</tr>
<tr>
<td>5</td>
<td>Connect to GeoServer UI and select Trac&amp;Loc process</td>
<td>The inputs displayed are similar.</td>
</tr>
</tbody>
</table>

- **Interfaces under tests**: Front-end / GeoServer WPS DescribeProcess

3.3.6 EO4WL_INT_TEST_015_FRONTEEND_EXECUTE_PROCESS

- **Test objective**: Execute a WPS process deployed on GeoServer through the Front-end access

- **Prerequisites**:
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END
  - EO4WL_INT_TEST_007_WEBSERVICES_FILE_MANAGER

- **Prerequisites**:
  - EO4WL_INT_TEST_006_DEPLOYMENT_FRONT_END
  - Initial data are available in the common workspace

- **Test steps**:
## System Integration and Validation Test Plan

### Description

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Through HTTP/REST access, call the webservice FileManagerServer with</td>
<td>Request returns the file list</td>
</tr>
<tr>
<td></td>
<td>- “List files” request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- on common workspace directory</td>
<td></td>
</tr>
</tbody>
</table>

- **Interfaces under tests:** Webservices / DFS
  - EO4WL_INT_TEST_008_WEBSERVICES_EXECUTE_PROCESS
  - EO4WL_INT_TEST_013_FRONTEND_GEOSERVER
  - EO4WL_INT_TEST_014_FRONTEND_GEOSERVER
  - A test process is deployed into GeoServer
  - Associated testing input data is available into test user workspace

### Test steps:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test user connects to platform from URL</td>
<td>Platform is accessible</td>
</tr>
<tr>
<td>2</td>
<td>User navigates to marketplace</td>
<td>EO4wildlife application is available</td>
</tr>
<tr>
<td>3</td>
<td>User selects EO4wildlife application</td>
<td>EO4wildlife home page is displayed</td>
</tr>
<tr>
<td>4</td>
<td>User selects Process execution page, and selects the process from list of</td>
<td>Process input form is displayed</td>
</tr>
<tr>
<td></td>
<td>available processes</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>User selects input data files from its user</td>
<td>Process is successfully executed</td>
</tr>
<tr>
<td></td>
<td>workspace and launches the execution</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User browses its workspace and download process outputs</td>
<td>Files are downloaded</td>
</tr>
</tbody>
</table>

- **Interfaces under tests:** Front end / Webservice ExecuteProcessService

### 3.3.7 EO4WL_INT_TEST_016_FRONTEND_PROCESS_STATUS

- **Test objective:** Monitor status of WPS process executions through Webservice through Front-end
- **Prerequisites:**
  - EO4WL_INT_TEST_009_WEBSERVICES_PROCESS_STATUS

### Test steps:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
<th>Expected result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User launches 3 process executions:</td>
<td>Processes are launched</td>
</tr>
<tr>
<td></td>
<td>- Successful process</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- One with invalid input that make execution fail</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- One successfully executed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- One in progress</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User selects execution status page</td>
<td>Status of the 3 processes are displayed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Successful process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Failed process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- In progress process</td>
</tr>
</tbody>
</table>

- **Interfaces under tests:** Front-end / Webservice ExecutionStatusService
4 Validation test cases

The validation test cases are described into dedicated deliverable Evaluation and validation Plan [2].
5 Conclusion

This document presents in a first step the process for the integration of EO4wildlife platform, including a definition of the roles for the different partners.

In a second step, this plan describes both the environments and the integration procedure which define the execution environment for the integration tests described in last section. These tests correspond to the design of EO4wildlife Platform v2 and an update of the design for Platform v3 could lead to a new edition of the current deliverable in order to take into account updates of the integration tests.
References

[1] D2.3 External interface for data discovery and processing v1.1, Deliverable of the EO4wildlife project, 2016


Annex I - Script integration detailed on Birdlife WPS service

The attached document describes the script integration steps from a R script to a WPS processing service for Birdlife script.

BirdLife Processing Service_v1.0.docx
From a Birdlife R script to processing service executable through WPS

Annex to D2.7 System Integration and Validation Test Plan
Table of Contents

1 Introduction ......................................................................................................................... 3
2 Processing service Input/Output handling ........................................................................ 4
  2.1 Tracking data .................................................................................................................. 4
  2.2 Auxiliary data ................................................................................................................ 4
3 Workflow - Publication and execution of a processing service into Geoserver .............. 5
4 Birdlife processing service ................................................................................................. 6
  4.1 Birdlife R script (step 1) ............................................................................................... 6
  4.2 Birdlife Docker file (step 2) .......................................................................................... 8
  4.3 Geoserver process definition (step 3) ........................................................................... 8
5 Birdlife processing services - Description, Execution, Status ........................................ 11
  5.1 List available processing services .................................................................................. 11
  5.2 Describe the Birdlife processing service ........................................................................ 11
  5.3 Execute asynchronously the Birdlife processing service (step 4) ................................ 12
  5.4 Get the status and results of the process ...................................................................... 12

List of Figures

Figure 1: R script updated before packaging into Docker ......................................................... 7
1 Introduction

This document introduces the publication and execution of a processing service into the platform and gives an overview of the complete workflow through an example: the integration of a subset of the Birdlife R script as a process runnable through WPS standard. It aims at testing the technical integration of services defined as an R script into GeoServer. The functional part of the service is then not considered in this document.

In a second section, it describes the packaging of a subpart of the provided Birdlife R script as a processing service into GeoServer. The pieces of code introduced in this document focus on the main parts useful for the integration of the processing service (shared directories, environment variables ...).

In a third section, it gives an overview of the common operations to list the available processing services, describe and execute a given processing service and retrieve the results of a process.

This document completes the section about processing services from D2.4 Platform Delivery v1 - User Guide v1.0.
2 Processing service Input/Output handling

This section details how the input and output will be handled by the processing services.

2.1 Tracking data

Tracking data will be uploaded by the user in his user workspace. In the process execution definition, a tracking data will then be defined with the path to the resource. The resource is accessible by Geoserver.

2.2 Auxiliary data

Auxiliary data are made available on the platform in a common directory accessible by all users. In the process execution definition, a raster data will then be defined with the path to the resource. The resource is accessible by Geoserver.
3 Workflow - Publication and execution of a processing service into Geoserver

1. Extract inputs (elements that depend on the process execution) from R script as environment variables
2. Package the R script into a Docker and push the Docker image into the Docker registry
   Note: also available in Python language but not used at the moment.
   a. Use the annotations to describe the process, inputs and outputs
   b. Implement method "execute" and set correct inputs/outputs. This method corresponds to what is launch through WPS Execute request
      i. Initialize execution variables and workspace as a subfolder of GeoServer exchange directory (/data/exchange/\langle user\rangle/\langle process dir\rangle/\langle execution ID\rangle). Define 4 subdirectories to organize the processing: .in, out, config, tmp
      ii. Launch JobScheduler:
         1. Partitioning according to input directories (optional, not used for the birdlife service processing)
         2. Docker execution(s) corresponding to script execution on each partition
         3. Merge resulting files from all partitions (optional, not used for the birdlife service processing)
      iii. Return results: return path to output data
4. Launch a GeoServer process using WPS standard
   a. make inputs available:
      i. Auxiliary data are made available in the common data directory
      ii. Tracking data are manually uploaded in the user workspace
   b. Generate WPS request with associated XML file and fill inputs in this file

Examples:

<table>
<thead>
<tr>
<th>For Primitive type:</th>
<th>For input File:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;wps:DataInputs&gt;</code></td>
<td><code>&lt;wps:DataInputs&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wps:Input&gt;</code></td>
<td><code>&lt;wps:Input&gt;</code></td>
</tr>
<tr>
<td><code>&lt;ows:Identifier&gt;</code></td>
<td><code>&lt;ows:Identifier&gt;</code></td>
</tr>
<tr>
<td>trackingDataFilesNb</td>
<td>trackingData</td>
</tr>
<tr>
<td><code>/wps:Input</code></td>
<td><code>/wps:Input</code></td>
</tr>
<tr>
<td><code>&lt;wps:Data&gt;</code></td>
<td><code>&lt;wps:Data&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wps:LiteralData&gt;</code></td>
<td><code>&lt;wps:LiteralData&gt;</code></td>
</tr>
<tr>
<td>1</td>
<td><code>/data/exchange/\langle user\rangle/input_dir/table.csv</code></td>
</tr>
<tr>
<td><code>/wps:Data</code></td>
<td><code>/wps:Data</code></td>
</tr>
<tr>
<td><code>&lt;wps:DataInputs&gt;</code></td>
<td><code>&lt;wps:DataInputs&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wps:Input&gt;</code></td>
<td><code>&lt;wps:Input&gt;</code></td>
</tr>
<tr>
<td><code>&lt;wps:DataInputs&gt;</code></td>
<td><code>&lt;wps:DataInputs&gt;</code></td>
</tr>
</tbody>
</table>
4 Birdlife processing service

The section illustrates the steps of the previous workflow with the integration of the Birdlife R script as a processing service.

4.1 Birdlife R script (step 1)

The script used for this testing purpose has been reduced. The script used for the testing computes then from a given input tracking data a shapefile corresponding to the 50% kernel utilization distribution (cf Lascelles_et_al-2016-Diversity_and_Distributions.pdf provided by Birdlife). It corresponds from step 1 to step 4 from Tracking Analysis processing described in the planned workflows defined by WP3.

Some minor changes have been done in order to deal with the processing service packaging later on. These changes include the use of environment variables to define the input files and directory (working dir) as well as the output folder. These parameters depend indeed on the execution of the processing service and need then to be set up at run time.

The path to the “Trackingdata_marineIBA_functions.r” script remains hard-coded. This file does indeed not depend on the execution of the process. It will then be part of the Docker (see section below for more details about the Docker definition and how this path is identified).

NB: Some other modification have been done concerning the longitude of the colony data frame (the previous value causes some failures) and the setting at the end of the output folder as the working directory (otherwise the result was saved into the working directory despite of the correct defined ‘td’ parameter).

The modification done have been highlighted in the script below.
`writeOGR` and `setwd` are functions in R for writing data to OGR-supported formats and setting the working directory, respectively. `tripSummary` and `DataGroupTrips` are data frames, and `names` is a function for extracting variable names. `for`, `if`, and `else` are control statements in R for looping and conditional evaluation. `plot` is a function for creating plots in R, and `head` is a function for viewing the first few rows of a data frame. `datagroupsUDd` and `Colony` are data frames, and `Colony$read.csv` is a function for reading a CSV file. `kpacks` and `new.packs` are packages, and `install.packages` is a function for installing R packages. `scaleARS` and `fpt.scales` are functions for data scaling and space-time partitioning, respectively. `tripSplit` and `subset` are functions for data manipulation. `DataGroup$Projected` and `DataGroup$Wgs` are data frames, and `CRS` and `proj4string` are functions for coordinate reference system handling. `data.frame` and `SpatialPointsDataFrame` are functions for creating data frames and spatial data frames, respectively. `temppath` and `tempdir` are functions for creating temporary paths or directories. `paste` and `paste0` are functions for string manipulation. `spatialPoints` is a function for creating spatial points data frames, and `strptime` is a function for parsing string time values. `writeOGR` is a function for writing data to OGR-supported formats.
4.2 Birdlife Docker file (step 2)

The Docker context includes the following files:
- The birdlife script defined in the previous section “birdlife.R”
- The “Trackingdata_marineIBA_functions.r” script
- The Docker file as below

```r
#Variables
#@Input(INPUT_FOLDER, working directory)
#@Input(TRACKING_DATA, file to the CSV Tracking data file)
#@Output(OUTPUT_FOLDER, Geoserver folder where resulting file are pushed)

#Base docker image in order to have R 3.3.1 available inside our container
FROM r-base:3.3.1

#Folder defined by ATOS FR to manage the data between the platform and the container
RUN mkdir -p /data/exchange
VOLUME ["/data/exchange"]

#Copy the birdlife.R and Trackingdata_marineIBA_functions.r
#into the /opt/scripts folder inside the docker container
COPY . /opt/scripts
WORKDIR /opt/scripts

#Install some additional R libraries required to execute the script
RUN apt-get update
RUN apt-get -y install libgeos-dev libgdal-dev tk-dev

#Command to be executed at the run of the container
CMD ["Rscript", "birdlife.R"]
```

Once built, the Docker image has to be pushed to the Docker registry to make it available for later use.

4.3 Geoserver process definition (step 3)

In order to define a custom processing service, Geoserver provides a java interface. This interface allows thanks to some annotation to

- describe the process
- describe the input parameters
- describe the output

Moreover it allows defining the processing to execute by a WPS Execute operation. For this purpose, the “Execute operation” needs to be implemented.

The implementation defined for the Birdlife tracking analysis process is introduced below. The execute method will be in charge of running the Docker container running the Birdlife R script. Before running the Docker container, the environment variables used in the Birdlife script need to be set up. The variables will then be set up as environment variable in the container when running the container.
The launching of Jobscheduler deals with the following steps:

1. Partitioning according to input directories (optional)
2. Docker execution(s) corresponding to script execution on each partition. The Docker container to be launched is defined in the podDefinitions method. The environment variables, volumes and docker image, previously pushed to the docker registry, are indeed set up. The piece of code below details the docker definition to be run by Jobscheduler.
3. Merge resulting files from all partitions (optional)

```java
@DescribeProcess(title = "birdlifetest", description = "compute the kernels for given input tracking data")
public class BirdlifeProcessing implements GeoServerProcess {

    @DescribeResult(name = "kernels", description = "output the shapefile resulting from the computation of kernels")
    public String execute(@DescribeParameter(name = "trackingData", description = "Path to the tracking data file") String trackingData, ProgressListener progressListener) {

        // some processing to be done when calling a WPS Execute operation
        // - Initialize execution variables and workspace
        Path inputFolder = IOTools.createFolder("in", exchangeFolder);
        //
        // - Launch JobScheduler by passing the parameters set up previously
        orderID = JSTools.addOrder(customClient, paramlist, BirdlifeConfiguration.JOBCHAINNAME);
        //
        // - Publish into Geoserver and return results
        return res;
    }
}
```
override protected def podDefinitions(parameters: JobParameters): List[Pod] = {
  val mountPoints = Seq(VolumeMount("exchangefolder",
    parametersdirectories.exchangeDir.toString))
  val volumes = Seq(HostVolume("exchangefolder",
    parametersdirectories.exchangeDir.toString))

  val trackingDataFilemame = new
  File(spooler_task.order.params().value("tracking_data")).getName

  val container = Container(
    name = "birdlife",
    image = "docker-registry/birdlife:latest",
    containerEnv = Seq(
      ContainerVar("TRACKING_DATA", trackingDataFilemame),
      ContainerVar("INPUT_FOLDER", partition.stagingPath.toString),
      ContainerVar("OUTPUT_FOLDER", partition.outputPath.toString),
      ContainerVar("ID_COMPUTATION", parameters.orderID)
    ),
    mountPoints = mountPoints
  )

  val metadata = this.podMetadata(parameters)

  List(Pod(
    metadata = metadata,
    containers = Seq(container),
    volumes = volumes,
    restartPolicy = Some("Never"),
    imagePullSecrets = Seq("key")
  ))
}

5 Birdlife processing services - Description, Execution, Status

5.1 List available processing services

As described in D2.4, the request below gives information about the WPS service and lists the processes available into Geoserver.

```
GET https://<geoserver>/geoserver/ows?service=WPS&version=1.0.0&request=GetCapabilities
```

5.2 Describe the Birdlife processing service

As described in D2.4, the request below describes the BirdlifeProcessing process, its input and output.

```
GET https://<geoserver>/geoserver/ows?service=WPS&version=1.0.0&request=DescribeProcess&Identifier=gs:BirdlifeProcessing
```

The response returned is the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessDescriptions xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en" service="WPS"
version="1.0.0" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 
http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
  <ProcessDescription wps:processVersion="1.0.0" statusSupported="true"
storeSupported="true">
    <ows:Identifier>gs:BirdlifeProcessing</ows:Identifier>
    <ows:Title>birdlifetest</ows:Title>
    <ows:Abstract>compute the kernels for given input tracking data</ows:Abstract>
    <DataInputs>
      <Input maxOccurs="1" minOccurs="1">
        <ows:Identifier>trackingData</ows:Identifier>
        <ows:Title>trackingData</ows:Title>
        <ows:Abstract>Path to the tracking data file</ows:Abstract>
        <LiteralData>
          <ows:AnyValue/>
        </LiteralData>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>kernels</ows:Identifier>
        <ows:Title>kernels</ows:Title>
        <LiteralOutput/>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</wps:ProcessDescriptions>
```
5.3 Execute asynchronously the Birdlife processing service (step 4)

The request below is an example of the execution of the Birdlife processing service in asynchronous mode with the input tracking data /data/exchange/eo4wildlife/user1/input_dir/table.csv. The upload service being not available on the platform yet, the input file has been manually uploaded on the platform. The request requires authentication.

```
POST https://<geoserver>/geoserver/ows?service=wps&version=1.0.0&request=Execute&Identifier=gs:BirdlifeProcessing

<?xml version="1.0" encoding="UTF-8"?>
<wps:Execute version="1.0.0" service="WPS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wcs="http://www.opengis.net/wcs/1.1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsAll.xsd">
  <ows:Identifier>gs:BirdlifeProcessing</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>trackingData</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>/data/exchange/eo4wildlife/user1/input_dir/table.csv</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:ResponseDocument status="true" storeExecuteResponse="true">
      <wps:RawDataOutput mimeType="application/zip">
        <ows:Identifier>kernels</ows:Identifier>
      </wps:RawDataOutput>
    </wps:ResponseDocument>
  </wps:ResponseForm>
</wps:Execute>
```

5.4 Get the status and results of the process

The request below gives the status of the BirdlifeProcessing process execution. The request requires authentication.

```
GET https://<geoserver>/geoserver/ows?service=WPS&version=1.0.0&request=GetExecutionStatus&executionId=c35be358-a770-40c7-bde9-075de83594c7
```

The executionId has to be substituted for the appropriate executionId retrieved from the response of the Execute operation described in the previous section.

The response of the previous request will depend of the current status of the process. It can be ProcessStarted, ProcessFailed, ProcessSucceeded.
In case of a successful process execution, you should get a response similar to the one below. The response returned the path to the resulting shapefile.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wps:ExecuteResponse xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xml:lang="en" service="WPS"
serviceInstance="https://<geoserver>/geoserver/ows?" statusLocation="https://<geoserver>/geoserver/ows?service=WPS&amp;version=1.0.0&amp;request=GetExecutionStat
us&amp;executionId=3b6bdc87-dee0-4b81-81f1-582b769a1568" version="1.0.0">
  <wps:Process wps:processVersion="1.0.0">
    <ows:Identifier>gs:BirdlifeProcessing</ows:Identifier>
    <ows:Title>birdlifetest</ows:Title>
    <ows:Abstract>compute the kernels for given input tracking data</ows:Abstract>
  </wps:Process>
  <wps:Status creationTime="2016-10-12T12:47:33.992Z">
  </wps:Status>
  <wps:ProcessOutputs>
    <wps:Output>
      <ows:Identifier>kernels</ows:Identifier>
      <ows:Title>output the shapefile resulting from the computation of kernels</ows:Title>
      <wps:Data>
        <wps:LiteralData>/data/exchange/eo4wildlife/user1/process_dir/BirdlifeID_YYYYMMDD_XX
X/testUD.shp</wps:LiteralData>
      </wps:Data>
    </wps:Output>
  </wps:ProcessOutputs>
</wps:ExecuteResponse>
```