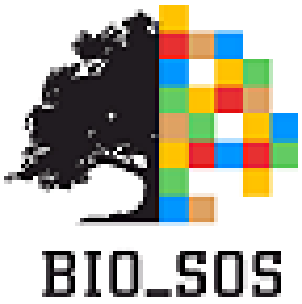

Project Title:	**BIO_SOS Biodiversity Multisource Monitoring System: from Space TO Species**
Contract No:	FP7-SPA-2010-1-263435
Instrument:	Collaborative Project
Thematic Priority:	FP7-SPACE-2010-1
Start of project:	1 December 2010
Duration:	36 months

Deliverable No: D.3.4

Interface Control Document

Due date of deliverable:

Actual submission date:

Version:1st version of D3.4

Main Authors: Dimitrios Karachalios (P13 - PKH); Diomede Iluzzi (P6 - PKI).

Project ref. number	**263435**
Project title	**BIO_SOS: Biodiversity Multisource Monitoring System: from Space to Species**

Deliverable title	ICD –Interface Control Document
Deliverable number	D3.4
Deliverable version	v1
Previous version(s)	
Contractual date of delivery	30/06/2013 (Month 31)
Actual date of delivery	04/07/2013
Deliverable filename	BIO_SOS_D3.4_ICD
Nature of deliverable	R = Report
Dissemination level	PU = Public
Number of pages	20
Workpackage	WP 3
Partner responsible	PKI
Author(s)	Dimitrios Karachalios (PKH) Diomede Illuzzi (PKI)
Editor	Diomede Illuzzi (PKI)
EC Project Officer	Florence Beroud

Abstract	The ICD describes the Interfaces of the system.
Keywords	System, architecture, workflow

Signatures

Written by	**Responsibility- Company**	**Date**	**Signature**
Dimitrios Karachalios	Technical Staff WP3 (PKH)	28/06/2013	
Diomede Illuzzi	Technical Staff WP3 (PKI)	28/06/2013	
Verified by			
Jens Stutte	WP3 Leader (PKI)	01/07/2013	
Jordi Inglada	WP5 Participant (USB)	05/07/2013	
Approved by			
Palma Blonda	Project Coordinator, CNR	05/07/2013	
Fifamè Koudogbo	Quality Team, AI	05/07/2013	

Table of Contents

1. Scope of the ICD document

This document provides detailed information for the interface definition in which each processor BIOSOS exposes functionality through the wrapper. The wrapper module is responsible for the exposure of the functionality of each local processor module BIO_SOS, such as web services on the internet. Principle concept of web service is the interface with which each web service is exposed by defining the communication and interaction of the service with the system. The ICD describes the inputs and outputs of a single BIO_SOS Processor and how to access the functions and services provided by a system via an interface.

1.1. EODHAM System General Considerations

The EODHAM System follows the concept of a Service-oriented architecture (SOA) . Purpose of SOA is to allow easy cooperation of a large number of computers that are connected over a network. Every computer can run an arbitrary number of programs. Called services in this context are built in a way that they can exchange information with any other service within the reach of the network without human interaction and without the need to make changes to the underlying program itself. The EODHAM system uses SOAP for the messages exchanges and WSDL for the description of the Interfaces. The work-flow of the system is orchestrating by the jBPM engine.

Follows abstract definitions of each concept that will help the technical comprehension. The ICD document focuses only on the WSDL definition that describes the interface of the services.

1.1.1. Simple Object Access Protocol (SOAP)

SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services (between client and server), in computer networks. It relies on XML Information Set for its message format, and usually relies on other Application Layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

1.1.2. Web Services Description Language (WSDL)

The Web Services Description Language is an XML-based interface description language that is used for describing the functionality offered by a web service. A WSDL description of a web service (also referred to as a WSDL file) provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. It thus serves a purpose that corresponds roughly to that of a method signature in a programming language.

1.1.3. Business Process Execution Language (BPEL)

Business Process Execution Language (BPEL), short for Web Services Business Process Execution Language (WS-BPEL) is an OASIS standard executable language for specifying actions within business processes with web services. Processes in BPEL export and import information by using web service interfaces exclusively. WS-BPEL provides a language for the specification of Executable and Abstract business processes. By doing so, it extends the Web Services interaction model and enables it to support business transactions. WS-BPEL defines an interoperable integration model that should facilitate the expansion of automated process integration both within and between businesses.

1.1.4. Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) is a standard for business process modelling that provides a graphical notation for specifying business processes in a Business Process Diagram (BPD),based on a flowcharting technique very similar to activity diagrams from Unified Modelling

Language (UML).[The objective of BPMN is to support business process management, for both technical users and business users, by providing a notation that is intuitive to business users, yet able to represent complex process semantics. The BPMN specification also provides a mapping between the graphics of the notation and the underlying constructs of execution languages, particularly Business Process Execution Language (BPEL).

1.1.5. Asynchronous and Synchronous Process

An Asynchronous Process is one which you call and need not wait for the response before proceeding further, when initiated doesn't reply back instantaneously. In our case the jBPM engine that controls the logic of the work-flow interaction can make asynchronous invocations to the involved BIO_SOS processor module web service. In that case the jBPM engine in the role of a client makes a request input message to the web service in an asynchronous way. That means that does not blocks the work-flow instance until the time that will receive the response from the web service (BIO_SOS Processor module). The BIO_SOS processor web service receives the input and begins the processing activity of the input data. It is expected that the processing activity is a time consuming process. At the end of the elaboration, the BIO_SOS processor web service returns the response to the requester as a separate exchanged message.

Synchronous processes are those which send their reply as the return message to the request instantaneously. For example in case of simulation scenario, based only to the meta-data and not to the data itself, where we need to extract some initial considerations of the global BIO_SOS processing capability, each processor can reply instantly considering that the processing in that case is only a simulation and needs less time to accomplish that process.

1.2. BIO_SOS Processor as web Service entity.

The wrapper module has the responsibility to expose the BIO_SOS processor functionality as web services. Wrapper generally refers to a type of packaging. Through the WSDL Interface permits the communication with the other services and activates the internal processing activities in demand of the input messages. After the generation of the output products, replies the response to the requester.

The Wrapper BIO_SOS Processor wrapper module is build with the JAX-WS 2.2.5 (Java API fro XML Web Services). With JAX-WS, Web services are called both synchronously and asynchronously. JAX-WS adds support for both a polling and callback mechanism when calling Web services asynchronously. Using a polling model, a client can issue a request; get a response object back, which is polled to determine if the server has responded. When the server responds, the actual response is retrieved. Using the callback model, the client provides a callback handler to accept and process the inbound response object. Both the polling and callback models enable the client to focus on continuing to process work without waiting for a response to return, while providing for a more dynamic and efficient model to invoke Web services.

2. WSDL Processor

The following xml is the WSDL for the Processor Interface that permits synchronous and asynchronous communication.

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions xmlns:tns="http://processing.biosos.eu" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://processing.biosos.eu"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
   <types>
      <xs:schema xmlns:tns="http://processing.biosos.eu"
         xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
         targetNamespace="http://processing.biosos.eu">

         <xs:element name="ProcessingException" nillable="true" type="xs:string"/>
         <xs:element name="describeMetaDataIn" nillable="true" type="tns:metaDataSet"/>
         <xs:element name="describeMetaDataOut" nillable="true" type="tns:metaDataSet"/>
         <xs:element name="executeMetaDataIn" nillable="true" type="tns:metaDataSet"/>
         <xs:element name="executeMetaDataOut" nillable="true" type="tns:metaDataSet"/>
         <xs:element name="metaDataSet" type="tns:metaDataSet"/>
         <xs:complexType name="metaDataSet">
            <xs:sequence>
               <xs:element name="metaDataList" type="xs:string" minOccurs="0"
                  maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:complexType>
      </xs:schema>
   </types>
   <message name="describeProcessing">
      <part name="describeMetaDataIn" element="tns:describeMetaDataIn"/>
   </message>
   <message name="describeProcessingResponse">
      <part name="describeMetaDataOut" element="tns:describeMetaDataOut"/>
   </message>
   <message name="ProcessingException">
      <part name="fault" element="tns:ProcessingException"/>
   </message>
   <message name="executeProcessing">
      <part name="executeMetaDataIn" element="tns:executeMetaDataIn"/>
   </message>
   <message name="executeProcessingResponse">
      <part name="executeMetaDataOut" element="tns:executeMetaDataOut"/>
   </message>
   <portType name="IProcessingService">
      <operation name="describeProcessing">
         <input xmlns:ns1="http://www.w3.org/2007/05/addressing/metadata"
            ns1:Action="http://processing.biosos.eu/IProcessingService/describeProcessingRequest"
            message="tns:describeProcessing"/>
         <output xmlns:ns2="http://www.w3.org/2007/05/addressing/metadata"
            ns2:Action="http://processing.biosos.eu/IProcessingService/describeProcessingResponse"
            message="tns:describeProcessingResponse"/>
         <fault xmlns:ns3="http://www.w3.org/2007/05/addressing/metadata"
            message="tns:ProcessingException" name="ProcessingException"
```

```
ns3:Action="http://processing.biosos.eu/IProcessingService/describeProcessing/Fault/ProcessingExcepti
on" />
      </operation>
      <operation name="executeProcessing">
         <input xmlns:ns4="http://www.w3.org/2007/05/addressing/metadata"
            ns4:Action="http://processing.biosos.eu/IProcessingService/executeProcessingRequest"
            message="tns:executeProcessing"/>
         <output xmlns:ns5="http://www.w3.org/2007/05/addressing/metadata"
            ns5:Action="http://processing.biosos.eu/IProcessingService/executeProcessingResponse"
            message="tns:executeProcessingResponse"/>
         <fault xmlns:ns6="http://www.w3.org/2007/05/addressing/metadata"
            message="tns:ProcessingException" name="ProcessingException"
ns6:Action="http://processing.biosos.eu/IProcessingService/executeProcessing/Fault/ProcessingExcepti
on" />
      </operation>
   </portType>

   <binding name="ProcessorUseCase1PortBinding" type="tns:IProcessingService">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="describeProcessing">
         <soap:operation soapAction="http://processing.biosos.eu/describeProcessing"/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
         <fault name="ProcessingException">
            <soap:fault use="literal" name="ProcessingException"/>
         </fault>
      </operation>
      <operation name="executeProcessing">
         <soap:operation soapAction="http://processing.biosos.eu/executeProcessing"/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
         <fault name="ProcessingException">
            <soap:fault use="literal" name="ProcessingException"/>
         </fault>
      </operation>
   </binding>
   <service name="x-usecase1">
      <port name="ProcessorUseCase1Port" binding="tns:ProcessorUseCase1PortBinding">
         <soap:address location="http://toyota.planetek.it:8080/jaxws-biosos/x-usecase1"/>
      </port>
   </service>
</definitions>
```

2.1. Graphical presentation of the WSDL

The WSDL is a descriptive document of the Interface. The following graphical presentations have the purpose to explain better the structure of the Interface for synchronous and asynchronous process.

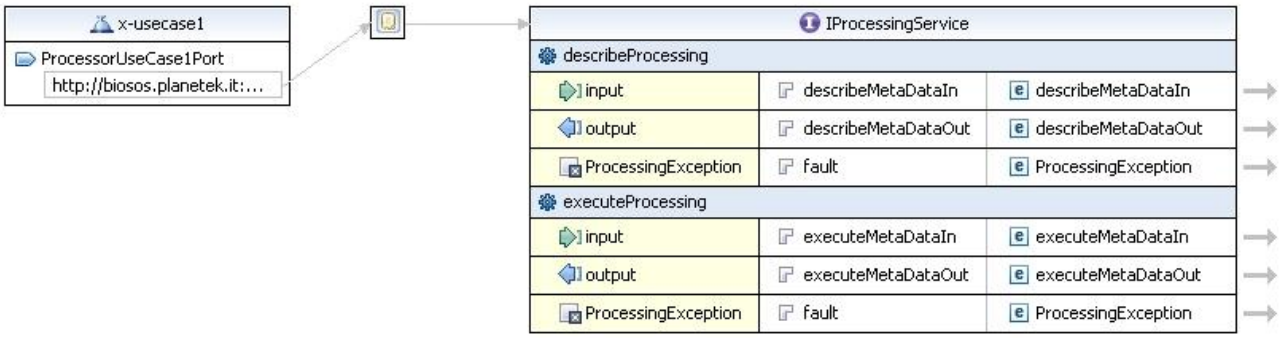

Figure 1 – The WSDL schema presentation of the processor service named x-usecase1

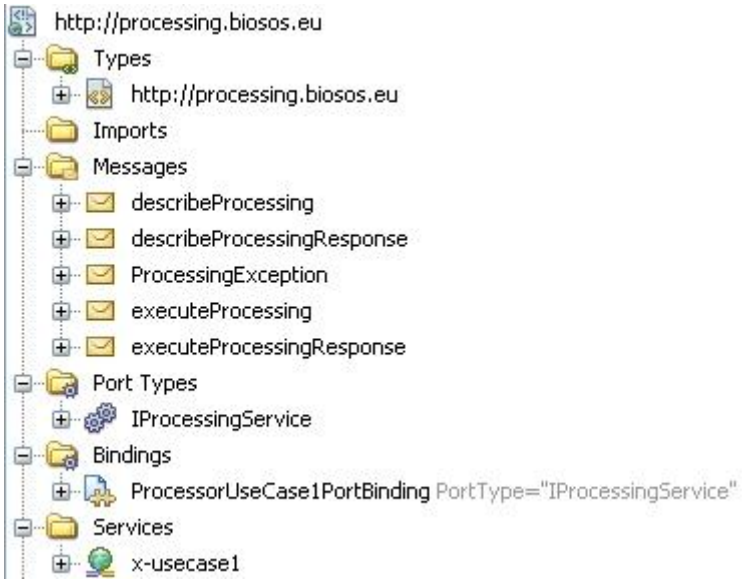

Figure 2 – Schematic structure of the WSDL

2.1.1. Target Namespace

```
<definitions xmlns:tns="http://processing.biosos.eu"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://processor1.biosos.eu"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
```

In the schema declaration the following fragment

xmlns:xsd=”http://www.w3.org/2001/XMLSchema” indicates that the elements and data types used in the schema come from the “http://www.w3.org/2001/XMLSchema” namespace.

The fragment targetNamespace="http://processing.biosos.eu” indicates that the elements defined by this schema come from the “http://processing.biosos.eu” namespace.

The fragment xmlns:tns="http://processing.biosos.eu" indicates that the elements and data types that come from the “http://processing.biosos.eu” namespace should be prefixed with tns.

It is good advising, each separate implementation of processor interface to use a unique value for the targetNamespace as for example http://processing1.biosos.eu , http//processing2.biosos.eu and so on.

2.1.2. Types

The <types> element is a container for data type definitions used by the web service. Our definitions use the following elements:

- **ProcessingException** an element of String type.
- **describeMetaDataIn** a complex element of metaDataSet type .
- **describeMetaDataOut** a complex element of metaDataSet type .
- **executeMetaDataIn** a complex element of metaDataSet type .
- **executeMetaDataOut** a complex element of metaDataSet type .
- **metaDataSet** element is an element of metaDataList type.
- **metaDataList** element is an element of String type.

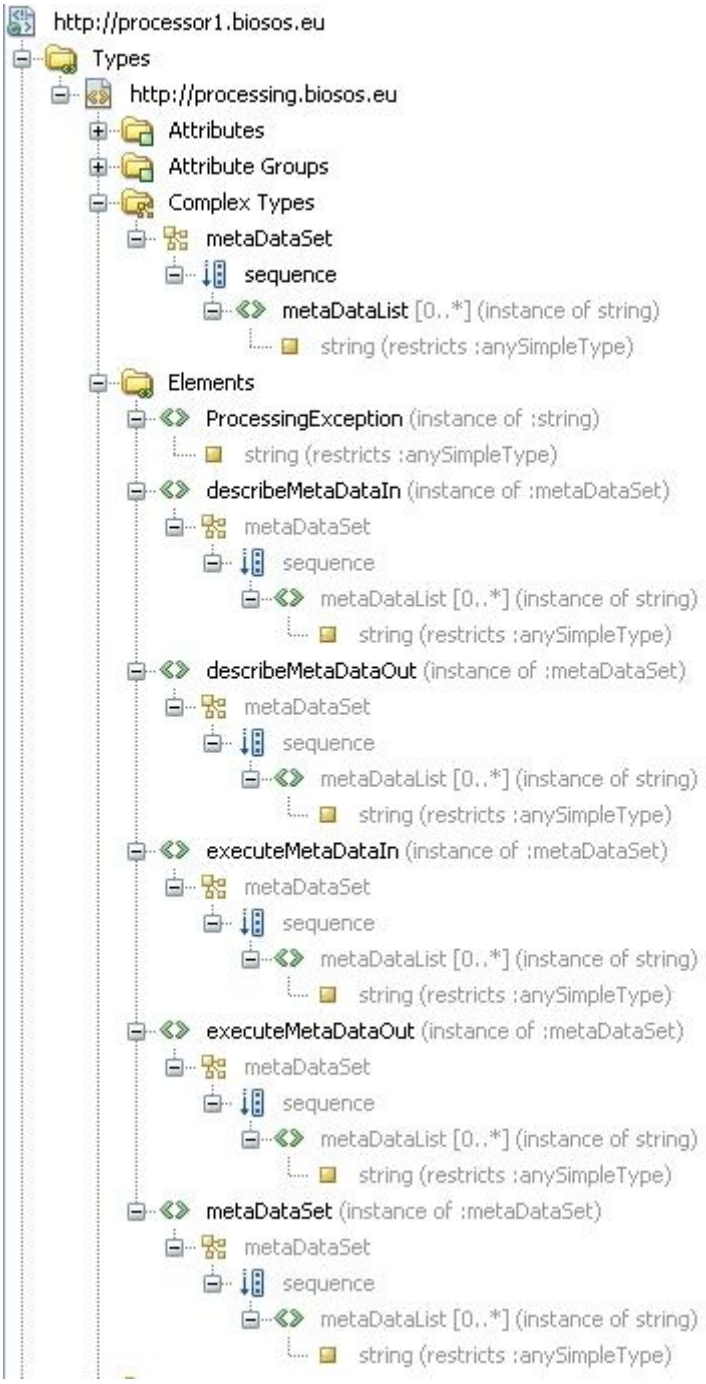

Figure 3 – Types of the WSDL

2.1.3. metaDataSet element

It is important to note that the metaDataSet is a list of metaDataList element of string type. That permits to create the input list of the metadata that have to be passed to the processor. Each metaDataList must contain the ftp url of the published metadata.xml

Example:

```
<metatns:describeMetaDataOut xmlns:metatns="http://processing.biosos.eu"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
   <metatns:metaDataList>
ftp://username:password@host:port/path/metadata.xml
</metatns:metaDataList>
   <metatns:metaDataList>
 ftp://username:password@host:port/path/metadata_2.xml
 </metatns:metaDataList>

    </metatns:describeMetaDataOut>
```

2.1.4. Message

A typed definition of the data being communicated:

- **describeProcessing** (PartName describeMetaDataIn , element thns:describeMetaDataIn)
- **describeProcessingResponse** (PartName describeMetaDataOut, element tns:describeMetaDataOut)
- **ProcessingException** (PartName fault, element tns:ProcessingException)
- **executeProcessing** (PartName executeMetaDataIn, element tns:executeMetaDataIn)
- **executeProcessingResponse** (PartName executeMetaDataOut, element tns:executeMetaDataOut)

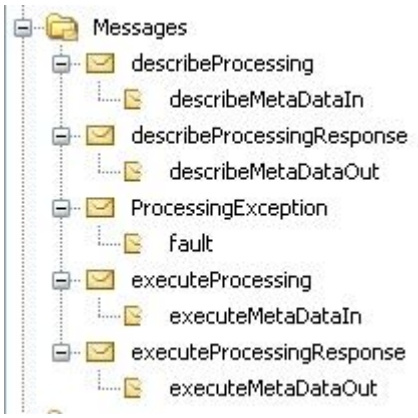

Figure 4 – Messages of the WSDL

2.1.5. portType

A set of operations supported by one or more endpoints

In the synchronous WSDL definition the IProcessingService support the following synchronous operations:

- **describeProcessing**
- **executeProcessing**

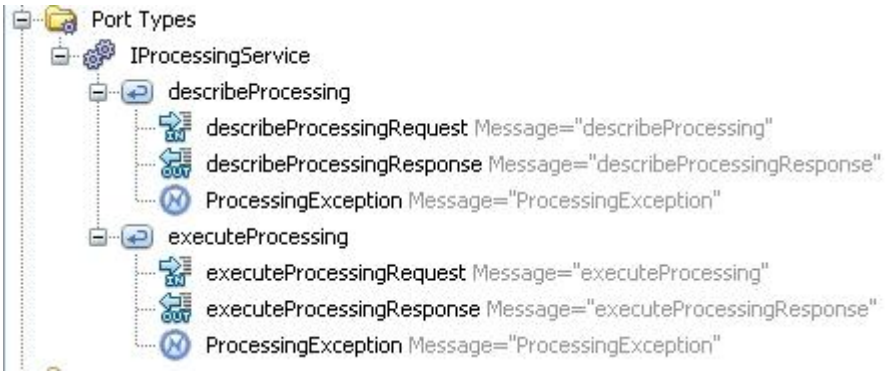

Figure 5 – Port types of WSDL

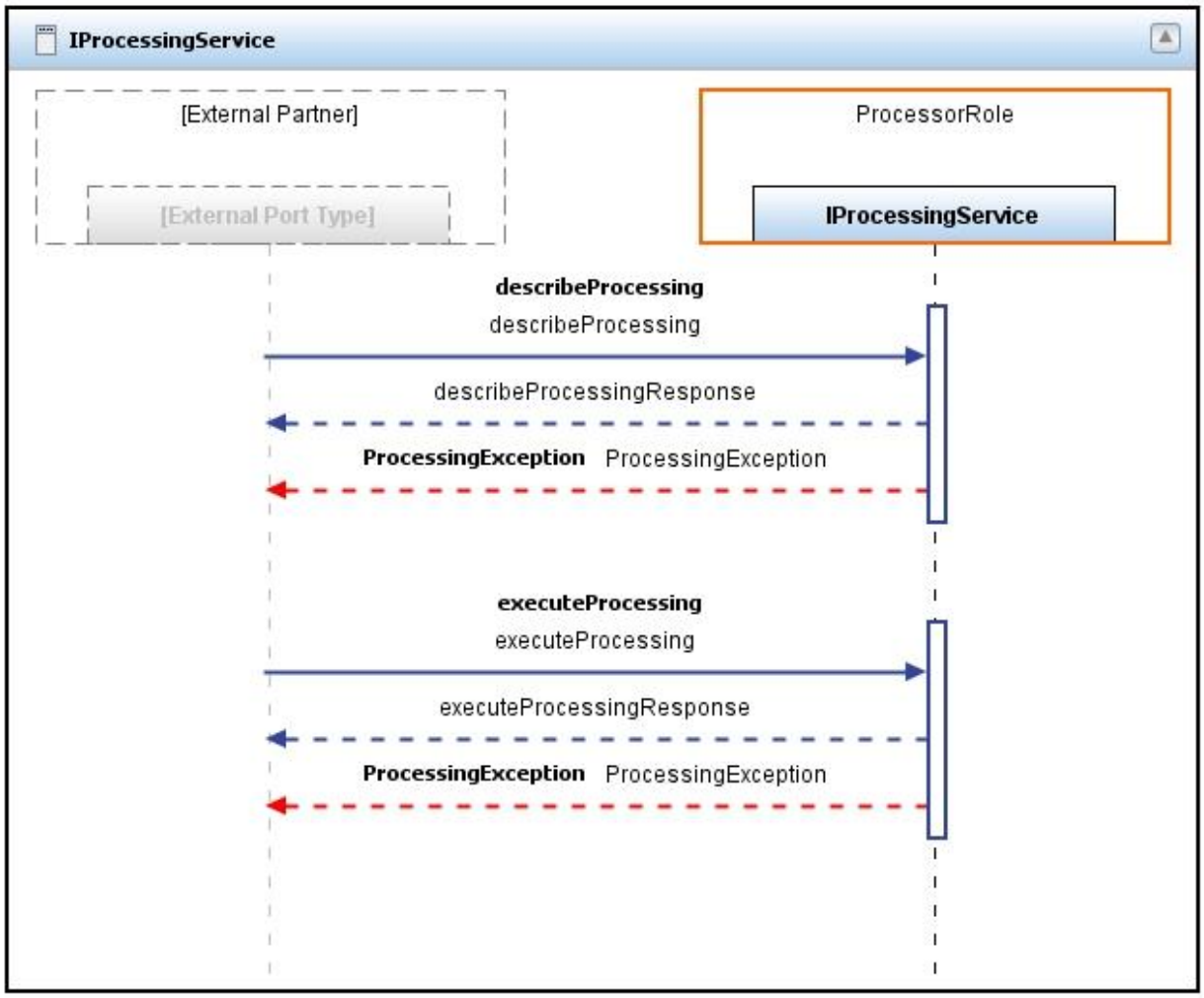

Figure 6 – Message exchange

Port Type : IProcessingService		
Operation Name	**Input message**	**Output message**
describeProcessing	describeProcessing	describeProcessingResponse
executeProcessing	executeProcessing	executeProcessingResponse

2.1.6. describeProcessing Operation

The EODHAM system permits a processing simulation of one or more BIO_SOS processors. In this operation the nature of the exchanged messages “describeMetaDataIn and describeMetaDataOut” are of the declared type metaDataSet .

MetaDataSet is a list of the element metaDataList. Each metaDataList has as value a FTP url path that points to an input metadata.xml. Through the execution of the BPEL work-flow one or more BIO_SOS processor web service is invoked using the interface Port type IProcessingService and the operation describeProcessing. The exchanged message is a list of metadata.xml files. The wrapper receives that input (a list of path of ftp ulrs) and proceeds with the download of the files. After that it executes a processing simulation based only on the information of the metadata files. In that operation, no real datasets are processed. The scope of the processing simulation is to response to the following question.

“What products can be obtained by the processor having as input that hypothetical data?”

During the describeProcessing operation the following activities have to be accomplished by the processor:

- The processor downloads the metadata files, found on the list content of the executeMetaDataIn element.
- The processor executes a processing simulation based on the info of the input metadata files.
- The processor creates the metadata files as response to that simulation.
- The processor uploads the metadata files that correspond to the hypothetical products to a specified repository.
- The processor replies to the workflow the describeMetaDataOut. The content of that message is a new list of metadata FTP ulr path that corresponds to the hypothetical products.

The descirbeProcessing operation in that case is considering that uses synchronous mode. In that case the reply of the Processor is considering that is giving back, almost as instantly response.

2.1.7. executeProcessing Operation

The executeProcessing Operation executes the real processing activity. The exchanged messages “executeMetaDataIn and executemetaDataOut” are also of the declared type metaDataSet. The difference in this operation is that the BIO_SOS web service has to download not only the metadata xml files, but also the dataset input files. The information of each necessary input dataset (a dataset FTP url) is included in the opposite metadata xml file.

During the executionProcessing operation the following activities have to accomplish by the BIO_SOS processor web service:

- The processor downloads the metadata files, found on the list content of the executeMetaDataIn element.
- The processor extracts from each metadata file the corresponding data FTP URL.
- The processor downloads the data input.
- The processor executes the processing activity and produces the corresponding products.
- The processor creates the metadata files for the produced products.
- The processor uploads the products to a specified repository.
- The processor uploads the metadata files that correspond to the products to a specified repository.
- The processor replies to the workflow the executeMetaDataOut. The content of that message is a new list of metadata FTP ulr path that corresponds to the products.

The executeProcessing operation in that case is considering that uses asynchronous mode. In that case the reply of the Processor is considering that is giving back, after a time period when all the internal processing activities have been completed.

2.1.8. Binding

A protocol and data format specification for a particular port type .

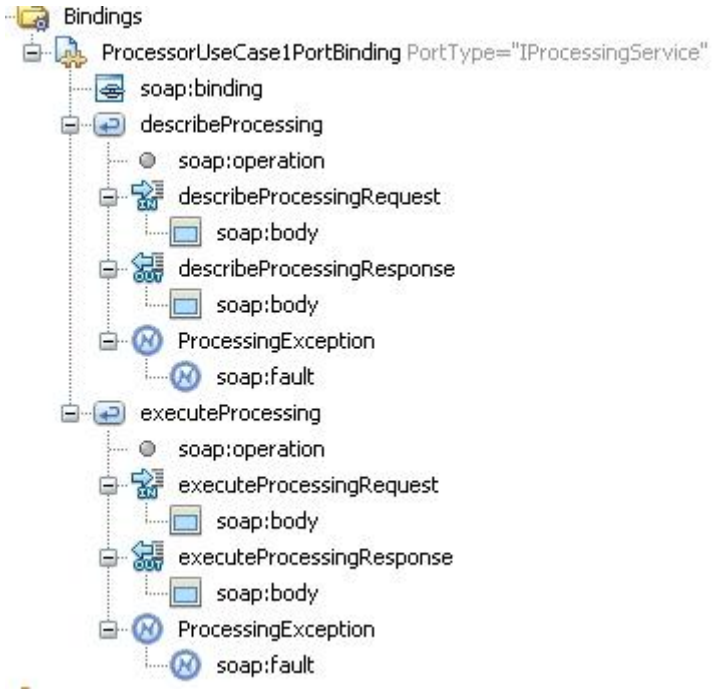

Figure 7 – Bindings of WSDL

Binding : ProcessorUseCase1PortBinding	
Port Type	IProcessingService
Extensibility	`<soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>`
Operations	describeProcessing, executeProcessing,

2.1.9. Services

Contains the endpoint address of the interface:

Figure 8 – Services of WSDL

It has to be unique for each web service.

3. References

1. Web Service,
   - http://en.wikipedia.org/wiki/Web_service
   - http://www.w3schools.com/webservices/ws_intro.asp

2. Web Services Business Process Execution Language (WS-BPEL) ,
   - http://docs.oasis-open.org/wsbpel/2.0/varprop
   - http://www.information-management.com/infodirect/20060602/1055720-1.html?zkPrintable=1&nopagination=1

3. Business Process Execution Language for Web Services (BPEL4WS),
   - http://xml.coverpages.org/bpel4ws.html

4. Web Services Description Language (WSDL) 1.1 ,
   - http://www.w3.org/TR/wsdl
   - http://www.w3.org/TR/ws-desc-usecases/
   - http://www.w3.org/TR/ws-arch/

5. Web Services Description Language (WSDL) Version 2.0,
   - http://www.w3.org/TR/wsdl20/

6. Simple Object Access Protocol,
   - http://en.wikipedia.org/wiki/SOAP
   - http://www.w3schools.com/soap/soap_intro.asp
   - http://www.w3.org/TR/ws-addr-soap/

7. Service-Oriented Architecture,
   - http://en.wikipedia.org/wiki/Service-oriented_architecture
   - http://msdn.microsoft.com/en-us/library/aa480021.aspx

8. Apache ODE, BPEL, Language Guide, WS-BPEL 2.0 Specification Compliance ,
   - http://ode.apache.org/ws-bpel-20-specification-compliance.html

9. Virtual Machine,
   - http://en.wikipedia.org/wiki/Virtual_machine

10. Jbpm
    - http://docs.huihoo.com/jboss/jbpm/5.4.0/html_single/#d0e481

11. BPMN
    - http://en.wikipedia.org/wiki/Business_Process_Model_and_Notation
    - http://www.omg.org/spec/BPMN/2.0/
      http://www.omg.org/spec/BPMN/2.0/PDF

4. Appendix I. Acronym and Abbreviation List

ADD	Architecture Design Document
API	Application Programming Interface
BIO_SOS	BIOdiversity multi-SOurce monitoring System: from Space TO Species
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execition Language for Web Services
BPM	Business Process Management
BPMN	Business Process Model and Notation
BIO_SOS	Biodiversity Multi-Source Monitoring System: From Space To Species
CEOS	Committee of Earth Observations
CERTH	Informatics And Telematics Institute Of The Centre For Research And Technology – Greece
CIBIO	Research Center in Biodiversity and Genetic Resources (Portugal)
CNR	Consiglio Nazionale delle Ricerche
DN	Digital Number
EO	Earth Observation
EODHaM	EO Data for Habitat Monitoring
EU	European Union
FP7	Seventh Framework Program
FTP	File Transfer Protocol
INSPIRE	Infrastructure for Spatial Information in Europe
ISO	International Organization for Standardization
JBoss	JavaBeans Open Source Software Application Server
JAX-WS	Java API for XML Web Services
ODE	Orchestration Director Engine
PKH	Planetek Hellas
PKI	Planetek Italia
QA4EO	Quality Assurance Framework for Earth Observation
QAP	Quality Assurance Plan
QI	Quality Indicator
RDS	Relational Database Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SDD	Service Design Document

SLA	Service Level Agreement
UDDI	Universal Description Discovery and Integration
WP	Work Package
WS	Web Service
WSDL	Web Service Description Language
XML	eXtensible Markup Language