EMSA CleanSeaNet
Data Centre
[CSN-DC]

Appendix 12 - EOP Lot1 Integration
Interface Control Document
[ICD]

Issue: 1.7

TABLE OF CONTENTS

LIST OF TABLES

1 INTRODUCTION

This is the Interface Control Document describing the CSN-DC external interfaces that have been expressely designed for EOP Lot 1 Integration. The document contains the detailed definition of the various interfaces, of the interface protocol and of the exchanged information.

1.1 DOCUMENT ORGANIZATION

The document defines the software interfaces between CSN-DC and the EOP Lot1 functionalities involved in any of the business processes.

The following sections are included in the document.

Section	Description
Introduction	This section
Interface definitions	A summary description of the new interfaces in terms of protocols and format to be used. The general schema for returned errors is also described
Query interfaces	A detailed description of all the new interfaces that will be exposed by CSNDC for query from EOP Lot1
Notification interfaces	
GIS Viewer Interfaces and configuration	
Performance and deployment	This chapter describe consideration about the new interfaces performance and systems deployment

1.2 REFERENCE DOCUMENTS

Document Title	Identifier	Internal Reference
EMSA Request for Services for Specific Contract n.4 within Framework Service Contract 15/EMSA/OP/17/2015: earth Observation data Centre (EODC) adaptation to Lot 1	EMSA.2016.026439	[RFO]
Multi communities multi sensors EO Processing Technical Specifications	v 2.0 dated 29/9/2015	[EMSA-TS]
EMSA SafeSeaNet Ecosystem - Request for Change – EODC – EODC adaptation to LOT1 – Business Requirements and Assessment of Technical Solution	V 1.0 dated 16/11/2016	[EMSA-RFC]
CSN-DC External Interface Control Document	CSNDC EICD v1.4.4	[CSN-EICD]

1.3 ABBREVIATIONS AND ACRONYMS

Abbreviation	Definition
AIS	Automatic Identification System
APT	Acquisition Planning Tool
BPD	Business Process Diagram
BPMN	Business Process Model Notation
CDM	Conceptual Data Model
COTS	Commercial Off The Shelf
CS	Coastal States
CSD	Clean Sea Net Service Desk
CSN-DC	Clean Sea Net Data Centre
DAIL	Data Access Interaction Layer
DREAM	Decision Support and Real Time EO Data Management
EO	Earth Observation
EOLI-SA	Earthnet On-Line Interactive – Stand Alone
ESA	European Space Agency
FEP	Front End Processor
GCM	GMES Contributing Mission
GMES	Global Monitoring for the Environment and Security
GML	Geographic Markup Language
GSCDA	GMES Space Component Data Access
GUI	Graphical User Interface
HMA	Heterogeneous Mission Accessibility
ICD	Interface Control Document
IF	Interface
IPF	Instrument Processing Facility
LRIT	Long Range Identification and Tracking
MSP	Model Service Provider
NRT	Near Real Time
OGC	Open Geospatial Consortium
OPeNDAP	Open-source Project for a Network Data Access Protocol
PKI	Public Key Infrastructure
SAR	Synthetic Aperture Radar
SO	Satellite Operators
SP	Service Providers

Abbreviation	Definition
SPA	Swath Planner Application
THREDDS	Thematic Realtime Environmental Distributed Data Services
UML	Unified Modelling Language
WFS	Web Feature Server
WMS	Web Map Server
XML	eXtensible Mark-up Language

Table 1-1 Abbreviations and Acronyms

2 INTERFACE DEFINITION

2.1 PROTOCOL & SECURITY

All the new interfaces will be made available using a REST format, using the HTTP protocol. If not specified, the HTTP/GET method is used.

Where foreseen, response data will be returned in JSON format.

The new interfaces will be available only within the CSN-DC (application-to-application) and will not be publicly accessible to CSN Users. Thus, no security is currently required or foreseen to be implemented (no SSL or Authentication).

2.2 ERROR MANAGEMENT

In case of errors, all the new interfaces will provide a spcific HTTP error (e.g. 404, 500).

Where applicable, a JSON object will be also returned, containing detailed information and messages describing the error condition, as specified in the following schema:

```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"Error schema",
   "type":"object",
   "additionalProperties":false,
   "required":[
      "code",
      "message",
      "description"
   ],
   "properties":{
      "code":{
         "type":"integer"
      },
      "message":{
         "type":"string"
      },
      "description":{
         "type":"string"
      }
   }
}
```

2.3 LIST OF INTERFACES

Name	Internal Module	URL Endpoint	Comment
GetPackageInfo	POR	http://[host]/por/rest/v1.0/package/[packageid]	Retrieves the packages info
GetOrdersList	POR	http://[host]/por/rest/v1.0/orders	Retrieves the list of pending orders expected by the CSN-DC
GetOrderDetails	POR	http://[host]/por/rest/v1.0/orders/[orderid]	Retrieves all information from an acquisition that is stored in the Planning for a given ID
GetAllOrganizations	POR	http://[host]/por/rest/v1.0/organisations	Retrieves all Organizations and if it is an "EMSA contractor"
GetAllOperations	POR	http://[host]/por/rest/v1.0/operations	Retrieves all Operations related to EODC
GetAllCountries	POR	http://[host]/por/rest/v1.0/countries	Retrieves all countries from the POR database
GetUsersList	POR	http://[host]/por/rest/v1.0/users	Retrieves the list of available users
GetUserInformation	POR	http://[host]/por/rest/v1.0/users/[userid]	Is an interface that given a userID it will return the list of Operations, Organizations and the Countries the user is associated with
GetPlatforms	FIN	http:// [host] /csndc-finsys-ws/rest/platforms?timestamp=<timestamp>	Retrieves the configured Platforms on the EMSA Class table
GetAllSensorModes	FIN	http:// [host] /csndc-finsys-ws/rest/sensorModes?platform=<platform>&timestamp=<timstamp>	Gets the sensor modes and respective polarizations configured for a specific Platform
GetServiceAlerts	OAS	http://[host]/oas/rest/v1.0/servicealerts/[serviceid]	Gets the user Alert Level of all Oil Spills available in an acquisition, by passing the acquisition identifier.
GetOilspillAlert	OAS	http://[host]/oas/rest/v1.0/oilspillalert/[oilspillid]	Gets the user Alert Level of an Oil Spill when passing the Oil Spill's event id

Table 2-1 CSN-DC List of Interfaces

Following sections will describe in details the above interfaces in terms of protocols, formats and operations.

3 QUERY INTERFACES

This chapter contains a detailed description of all the new interfaces that will be exposed by CSNDC for query from EOP Lot 1.

3.1 GETPACKAGEINFO

Retrieves the packages that are expected from a Service Provider for a given ID. The returned payload is a list of Providers, and for each Provider a list of package type and package name is provided. The package name is provided only if it was already announced to the CSN-DC system, through the Hash Service.

The list of expected packages is determined according to the Service Elements that are associated to the Service Type assigned to the SP/SO providers for the service.

The correspondence between Service Elements and Package_Types is the following:

Service Element	Package Type
ACTIVITY_DETECTION	ACTIVITY_DETECTION
CHANGE_DETECTION	CHANGE_DETECTION
DOWNLINK	EO_PRODUCT
IMAGE_DELIVERY	EO_PRODUCT, QUALITY_NOTIFICATION
IMAGE_PROCESSING	EO_PRODUCT
LICENSE	LIC
OIL_SPILL_DETECTION	OS_NOTIFICATION
OIL_SPILL_WARNING	OS_WARNING
OPTICAL_24H	EO_PRODUCT
OPTICAL_3H	EO_PRODUCT
OPTICAL_6H	EO_PRODUCT
OPTICAL_NRT	EO_PRODUCT
ORTHO	EO_PRODUCT
QUALITY_REPORT	QUALITY_REPORT
SAR_DERIVED	SAR_DERIVED
VDS_VESSEL_DETECTION	VESSEL_DETECTION
VESSEL_DETECTION	SAR_DERIVED

3.1.1 Protocol details

3.1.1.1 Request

Type	REST

URL	`http://[host]/por/rest/v1.0/packages/[packageid]`
Description	Returns a JSON object containing the information of the expected package
Method	HTTP GET
Parameters	• packageid: the service id of the expected package
Headers	NONE
Body	N/A
Schema	N/A

3.1.1.2 Response

Headers	NONE
Body	JSON object describing the packages info
Schema	

```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"GetPackageInfo Response",
   "oneOf":[
      {
         "type":"object",
         "additionalProperties":false,
         "required":[
            "satellite",
            "sensorMode",
            "serviceStatus",
            "projects",
            "providers"
         ],
         "properties":{
            "satellite":{
               "type":"string"
            },
            "sensorMode":{
               "type":"string"
            },
            "serviceStatus":{
               "type":"string"
            },
            "projects":{
               "type":"array",
               "items": {
                  "type": "string"
               }
            },
            "providers":{
               "type":"array",
               "items":{
                  "type":"object",
                  "required":[
                     "provider",
                     "packages"
                  ],
                  "additionalProperties":false,
                  "properties":{
```

```
                        "provider":{
                           "description":"Name of the provider",
                           "type":"string"
                        },
                        "packages":{
                           "description":"Array of foreseen
packages",
                           "type":"array",
                           "items":{
                              "type":"object",
                              "required":[
                                 "type",
                                 "name",
                                 "serviceElement"
                              ],
                              "additionalProperties":false,
                              "properties":{
                                 "type":{
                                    "description":"type of the
foreseen package",
                                    "type":"string",
                                    "enum":[
                                       "OS_WARNING",
                                       "EO_PRODUCT",
                                       "OS_NOTIFICATION",
                                       "SAR_DERIVED",
                                       "QUALITY_NOTIFICATION",
                                       "QUALITY_REPORT",
                                       "PROCESS_REQUEST",
                                       "PROCESS_RESPONSE",
                                       "ACTIVITY_DETECTION",
                                       "VESSEL_DETECTION",
                                       "CHANGE_DETECTION",
                                       "UNDEFINED"
                                    ]
                                 },
                                 "name":{
                                    "description":"file name",
                                    "type":["string", "null"]
                                 },
                                 "serviceElement":{
                                    "description":"the service
element for this package",
                                    "type":"string"
                                 }
                              },
                              "minItems":1
                           }
                        }
                     }
                  }
               }
            },
            {
               "type":"object",
               "additionalProperties":false,
               "required":[
```

	"code",
	"message",
	"description"
	],
	"properties":{
	"code":{
	"type":"integer"
	},
	"message":{
	"type":"string"
	},
	"description":{
	"type":"string"
	}
	}
	}
	]
	}

HTTP codes	• 200 OK: package information found • 400 Bad Request: non numeric id was requested • 404 Not Found: requested packages not found • 500 Internal Server Error: an unexpected backend error occurred

3.1.2 GetPackageInfo example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/packages/5622
```

Response:


```
{
   "satellite":"SENTINEL-1",
   "sensorMode":"IWS",
   "serviceStatus":"Delivered",
   "projects":[
      "CleanSeaNet"
   ],
   "providers":[
      {
         "provider":"ESA",
         "packages":[
            {
               "type":"UNDEFINED",
               "name":null,
               "serviceElement":"LICENSE"
            },
            {
               "type":"EO_PRODUCT",

"name":"1703140009_S1A_IW_GRDM_1SVV_20170314T055219_20170314T055357_015684_019C
EF_0C46_EOP.tgz",
               "serviceElement":"DOWNLINK"
            }
         ]
      },
```

```
      {
         "provider":"EGEOS",
         "packages":[
            {
               "type":"EO_PRODUCT",

"name":"1703140009_S1A_IW_GRDM_1SVV_20170314T055219_20170314T055357_015684_019C
EF_0C46_EOP.tgz",
               "serviceElement":"IMAGE_DELIVERY"
            },
            {
               "type":"QUALITY_NOTIFICATION",

"name":"1703140009_S1A_IW_GRDM_1SVV_20170314T055219_20170314T055357_015684_019C
EF_0C46_QNO.tgz",
               "serviceElement":"IMAGE_DELIVERY"
            },
            {
               "type":"SAR_DERIVED",
               "name":null,
               "serviceElement":"VESSEL_DETECTION"
            }
         ]
      }
   ]
}
```


3.2 GETORDERSLIST

Retrieves the list of all the orders in CSN-DC with Status >= Tasked (i.e. Tasked, Cancelled, Delivered, Error, Anomaly Conflict), where the status has been updated starting from a specific date (default: current date – 7 days).

3.2.1 Protocol details

3.2.1.1 Request

Type	REST			
URL	`http://[host]/por/rest/v1.0/orders`			
Description	Retrieves the list of expected orders, giving a small amount of information for each order			
Method	HTTP GET			
Parameters	startdate	String (format: YYYY-MM-DDTHH24:MM:SSZ)	Optional (default: current date – 7 days)	Used to limit the number of returned results
Headers	NONE			

Body	N/A
Schema	N/A

3.2.1.2 Response

Headers	NONE
Body	Json object with the orders list
Schema
```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"GetOrdersList Response",
   "oneOf":[
      {
         "type":"array",
         "items":{
            "type":"object",
            "required":[
               "orderId",
               "dto",
               "status",
               "lastUpdate",
               "platform",
               "product"
            ],
            "additionalProperties":false,
            "properties":{
               "orderId":{
                  "type":"integer"
               },
               "dto":{
                  "type":"integer"
               },
               "status":{
                  "type":"string"
               },
               "lastUpdate":{
                  "type":"string",
                  "pattern":" ^[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-
9]{2}:[0-9]{2}:[0-9]{2}Z$ "
               },
               "platform":{
                  "type":"object",
                  "description":"Acquisition platform
information",
                  "required":[
                     "satellite",
                     "sensor"
                  ],
                  "additionalProperties":false,
                  "properties":{
                     "satellite":{
                        "type":"string"
                     },
                     "sensor":{
                        "type":"object",
                        "required":[
                           "sensor",
```
 |

```
                              "mode"
                           ],
                           "additionalProperties":false,
                           "properties":{
                              "sensor":{
                                 "type":"string"
                              },
                              "mode":{
                                 "type":"string"
                              }
                           }
                        }
                     }
                  },
                  "product":{
                     "type":"object",
                     "description":"Product related information",
                     "required":[
                        "acquisitionStart",
                        "acquisitionStop",
                        "processingLevel"
                     ],
                     "additionalProperties":false,
                     "properties":{
                        "acquisitionStart":{
                           "type":"string",
                           "pattern":"^[0-9]{4}-[0-9]{2}-[0-
9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z$"
                        },
                        "acquisitionStop":{
                           "type":"string",
                           "pattern":"^[0-9]{4}-[0-9]{2}-[0-
9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z$"
                        },
                        "processingLevel":{
                           "type":"string"
                        }
                     }
                  }
               }
            }
         },
         {
            "type":"object",
            "additionalProperties":false,
            "required":[
               "code",
               "message",
               "description"
            ],
            "properties":{
               "code":{
                  "type":"integer"
               },
               "message":{
                  "type":"string"
               },
               "description":{
```

	"type":"string" } } }] }
HTTP codes	• 200 OK: order information found • 500 Internal Server Error: an unexpected backend error occurred

3.2.2 GetOrdersList example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/orders?startdate=2017-07-11T23:59:59Z
```

Response:


```
[

    {
        "orderId": 1110170000,
        "dto": 1,
        "status": "Tasked",
        "lastUpdate": "2017-07-12T17:30:10Z",
        "platform": {
            "satellite": "RADARSAT-2",
            "sensor": {
                "sensor": "SCNA",
                "mode": "SCNA"
            }
        },
        "product": {
            "acquisitionStart": "2011-10-17T17:07:17Z",
            "acquisitionStop": "2011-10-17T17:08:02Z",
            "processingLevel": "N/A"
        }
    }

]
```


3.3 GETORDERDETAILS

Retrieves all information from an acquisition that is stored in the Planning for a given ID. The satellite, sensor, acquisition start and stop dates, the planned footprint (in wkt format) and other important information are returned.

3.3.1 Protocol details

3.3.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/orders/[orderid]`
Description	Retrieves all information from an acquisition that is stored in the Planning for a given ID.
Method	HTTP GET
Parameters	• orderid: the id of the inspected order
Headers	NONE
Body	N/A
Schema	N/A

3.3.1.2 Response

Headers	NONE
Body	JSON object describing the order info
Schema
```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"GetOrderDetails Response",
   "oneOf":[
      {
         "type":"array",
         "items":{
            "type":"object",
            "additionalProperties":false,
            "properties":{
               "order":{
                  "type":"object",
                  "description":"Planning order information",
                  "required":[
                     "id",
                     "dto",
                     "lastUpdate",
                     "status",
                     "cart",
                     "region",
                     "serviceType",
                     "serviceDescription",
                     "flexibleArea",
                     "flexibleTime",
                     "maxDownlinkDelayInMinutes",
                     "maxDeliveryDelayServiceProvider1InHours",
                     "maxDeliveryDelayServiceProvider2InHours",
                     "serviceElementsServiceProvider1",
                     "serviceElementsServiceProvider2",
                     "planOperations",
                     "serviceProvider1",
                     "serviceProvider2"
```
 |

```
                    ],
                    "additionalProperties":false,
                    "properties":{
                       "id":{
                          "type":"integer",
                          "minimum":0
                       },
                       "dto":{
                          "type":"integer",
                          "minimum":0
                       },
                       "lastUpdate":{
                          "type":"string",
                          "pattern":"^[0-9]{4}-[0-9]{2}-[0-
9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z$"
                       },
                       "status":{
                          "type":"string",
                          "enum":[
                             "Idle",
                             "Ready for Confirmation",
                             "Confirmed",
                             "Not Confirmed",
                             "Ready for Allocation",
                             "Allocated",
                             "Not Allocated",
                             "Allocation Expired",
                             "Approval",
                             "Tasked",
                             "Cancelled",
                             "Delivered",
                             "Error",
                             "Anomaly"
                          ]
                       } ,
                       "cart":{
                          "type":"object",
                          "properties": {
                                "identifier":{
                                      "type":"integer"
                                },
                                "name":{
                                      "type":"string"
                                }
                          }
                       },                          "region":{
                          "type":"string"
                       },
                       "serviceType":{
                          "type":"string"
                       },
                       "serviceDescription":{
                          "type":"string"
                       },
                       "flexibleArea":{
                          "type":"boolean"
                       },
                       "flexibleTime":{
```

```
                        "type":"boolean"
                     },
                     "maxDownlinkDelayInMinutes":{
                        "type":"integer"
                     },
                     "maxDeliveryDelayServiceProvider1InHours":{
                        "type":"integer"
                     },
                     "maxDeliveryDelayServiceProvider2InHours":{
                        "type":"integer"
                     },
                     "serviceElementsServiceProvider1":{
                        "type":"array",
                        "items":{
                           "type":"string"
                        }
                     },
                     "serviceElementsServiceProvider2":{
                        "type":"array",
                        "items":{
                           "type":"string"
                        }
                     },
                     "planOperations":{
                        "type":"array",
                        "items":{
                           "type":"string"
                        }
                     },
                     "serviceProvider1":{
                        "type":"string"
                     },
                     "serviceProvider2":{
                        "type":"string"
                     }
                  }
               },
               "platform":{
                  "type":"object",
                  "description":"Acquisition platform
information",
                  "required":[
                     "satellite",
                     "sensor"
                  ],
                  "additionalProperties":false,
                  "properties":{
                     "satellite":{
                        "type":"string"
                     },
                     "sensor":{
                        "type":"object",
                        "required":[
                           "sensor",
                           "mode",
                           "typology",
                           "resolution",
                           "polarisation",
```

```
                        "polarisationMode",
                        "antennaLook"
                     ],
                     "additionalProperties":false,
                     "properties":{
                        "sensor":{
                           "type":"string"
                        },
                        "mode":{
                           "type":"string"
                        },
                        "typology":{
                           "type":"string"
                        },
                        "resolution":{
                           "type":"string"
                        },
                        "polarisation":{
                           "type":"string"
                        },
                        "polarisationMode":{
                           "type":"string"
                        },
                        "antennaLook":{
                           "type":"string",
                           "enum":[
                              "LEFT",
                              "RIGHT"
                           ]
                        }
                     }
                  }
               }
            },
            "product":{
               "type":"object",
               "description":"Product related information",
               "required":[
                  "footprint",
                  "area",
                  "acquisitionStart",
                  "acquisitionStop",
                  "processingLevel"
               ],
               "additionalProperties":false,
               "properties":{
                  "footprint":{
                     "type":"string"
                  },
                  "area":{
                     "description":"Area of the footprint,
expressed in square km. Null if geometry is not valid",
                     "type":[
                        "number",
                        "null"
                     ]
                  },
                  "acquisitionStart":{
```

<table>
<tr><td></td><td><pre>
                        "type":"string",
                        "pattern":"^[0-9]{4}-[0-9]{2}-[0-
9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z$"
                     },
                     "acquisitionStop":{
                        "type":"string",
                        "pattern":"^[0-9]{4}-[0-9]{2}-[0-
9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z$"
                     },
                     "processingLevel":{
                        "type":[“string”, “null”]
                     }
                  }
               }
            }
         }
      },
      {
         "type":"object",
         "additionalProperties":false,
         "required":[
            "code",
            "message",
            "description"
         ],
         "properties":{
            "code":{
               "type":"integer"
            },
            "message":{
               "type":"string"
            },
            "description":{
               "type":"string"
            }
         }
      }
   ]
}
</pre></td></tr>
<tr><td>HTTP codes</td><td>

200 OK: order information found
400 Bad Request: non numeric id was requested
404 Not Found: order not found
500 Internal Server Error: an unexpected backend error occurred

</td></tr>
</table>

3.3.2 GetOrderDetails example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/orders/5622
```

Response:

```
[
   {
```

```
      "order":{
         "id":5622,
         "dto":1,
         "lastUpdate":"2014-11-24T13:29:30Z",
         "status":"Tasked",
         "cart":{
            "identifier":1234,
            "name":"visibilta 2
         },
         "region":"Planning area 2",
         "serviceType":"SO Automatic",
         "serviceDescription":"",
         "flexibleArea":false,
         "flexibleTime":false,
         "maxDownlinkDelayInMinutes":0,
         "maxDeliveryDelayServiceProvider1InHours":0,
         "maxDeliveryDelayServiceProvider2InHours":0,
         "serviceElementsServiceProvider1":[
            "Data Downlink",
            "Image Delivery",
            "Image Processing",
            "License"
         ],
         "serviceElementsServiceProvider2":[
            "Oil Spill Detection",
            "Vessel Detection"
         ],
         "planOperations":[
            "test-operation"
         ],
         "serviceProvider1":"ESA",
         "serviceProvider2":"CLS"
      },
      "platform":{
         "satellite":"ENVISAT",
         "sensor":{
            "sensor":"ASAR",
            "mode":"WS",
            "typology":"RADAR",
            "resolution":"",
            "polarisation":"VV",
            "polarisationMode":"S",
            "antennaLook":"LEFT"
         }
      },
      "product":{
         "footprint":"MULTIPOLYGON (((9.242817 57.69753, 8.969466 57.14502,
8.704161 56.591938, 8.446514 56.0383, 8.187063 55.484264, 7.9408655 54.92962,
7.69393 54.374573, 7.4535847 53.819042, 7.219533 53.26303, 6.9914904 52.70658,
6.7633257 52.15046, 6.545588 51.59329, 6.3259315 51.03665, 6.1255555 50.516567,
0.4458943 51.246845, 0.5830705 51.770035, 0.7335614 52.33021, 0.8762431
52.88984, 1.0304447 53.449516, 1.1792203 54.008724, 1.3320485 54.567753,
1.4891255 55.126583, 1.650661 55.685207, 1.807622 56.243435, 1.9767685 56.80159,
2.1396806 57.359344, 2.3076196 57.916878, 2.4808543 58.47418, 9.242817
57.69753)))",
         "area":null,
         "acquisitionStart":"2011-03-30T10:06:50Z",
         "acquisitionStop":"2011-03-30T10:08:53Z",
```

```
            "processingLevel":"1A"
        }
    }
]
```

3.4 GETALLORGANIZATIONS

Retrieves the list of all Organizations and if it is an "EMSA contractor"

3.4.1 Protocol details

3.4.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/organisations`
Description	Retrieves all Organizations and if it is an "EMSA contractor"
Method	HTTP GET
Parameters	NONE
Headers	NONE
Body	N/A
Schema	

3.4.1.2 Response

Headers	NONE
Body	JSON object describing the organisations info

Schema	{
	"$schema":"http://json-schema.org/draft-04/schema#",
	"title":"GetAllOrganisations Response",
	"oneOf":[
	{
	"type":"array",
	"items":{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"organisation",
	"oimCode",
	"emsaContractor",
	"authority"
	],
	"properties":{
	"organisation":{
	"type":"string"
	},
	"oimCode":{

```
                    "type":"string"
                },
                "emsaContractor":{
                    "type":"boolean"
                },
                "authority":{
                    "type":"object",
                    "additionalProperties":false,
                    "required":[
                        "name",
                        "typology",
                        "code"
                    ],
                    "properties":{
                        "name":{
                            "type":"string"
                        },
                        "typology":{
                            "type":"string"
                        },
                        "code":{
                            "type":"string"
                        }
                    }
                }
            }
        },
        {
            "type":"object",
            "additionalProperties":false,
            "required":[
                "code",
                "message",
                "description"
            ],
            "properties":{
                "code":{
                    "type":"integer"
                },
                "message":{
                    "type":"string"
                },
                "description":{
                    "type":"string"
                }
            }
        }
    ]
}
```


HTTP codes	• 200 OK • 500 Internal Server Error: an unexpected backend error occurred

3.4.2 GetAllOrganizations example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/organisations
```

Response:


```
[
   {
      "organisation":"ESA",
      "oimCode":"XI01001",
      "emsaContractor":true,
      "authority":{
         "name":"EMSA Contractor",
         "typology":"COMPANY",
         "code":"XI"
      }
   },
   {
      "organisation":"MDA",
      "oimCode":"XI01006",
      "emsaContractor":true,
      "authority":{
         "name":"EMSA Contractor",
         "typology":"COMPANY",
         "code":"XI"
      }
   },
   {
      "organisation":"KSAT",
      "oimCode":"XI02004",
      "emsaContractor":true,
      "authority":{
         "name":"EMSA Contractor",
         "typology":"COMPANY",
         "code":"XI"
      }
   },
   {
      "organisation":"Not defined (AL)",
      "oimCode":"AL00000",
      "emsaContractor":false,
      "authority":{
         "name":"Albania",
         "typology":"NONEUCOUNTRY",
         "code":"AL"
      }
   },
   {
      "organisation":"Not defined (AT)",
      "oimCode":"AT00000",
      "emsaContractor":false,
      "authority":{
         "name":"Austria",
         "typology":"MEMBERSTATE",
         "code":"AT"
      }
   },
```

```
    {
        "organisation":"Not defined (BE)",
        "oimCode":"BE00000",
        "emsaContractor":false,
        "authority":{
            "name":"Belgium",
            "typology":"MEMBERSTATE",
            "code":"BE"
        }
    }
]
```


3.5 GETALLOPERATIONS

Retrieves all Operations related to EODC.

3.5.1 Protocol details

3.5.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/operations`
Description	Retrieves all operations related to EODC
Method	HTTP GET
Parameters	NONE
Headers	NONE
Body	N/A
Schema	N/A

3.5.1.2 Response

Headers	NONE
Body	JSON object describing the list of operations available
Schema	`{ "$schema":"http://json-schema.org/draft-04/schema#", "title":"GetAllOperations Response", "oneOf":[ { "type":"array", "items":{ "type":"string" } }, {`

<table>
<tr><td></td><td><pre>
"type":"object",
"additionalProperties":false,
"required":[
"code",
"message",
"description"
],
"properties":{
"code":{
"type":"integer"
},
"message":{
"type":"string"
},
"description":{
"type":"string"
}
}
}
]
}
</pre></td></tr>
<tr><td>HTTP codes</td><td>• 200 OK: package information found
• 500 Internal Server Error: an unexpected backend error occurred</td></tr>
</table>

3.5.2 GetAllOperations example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/operations
```

Response:

```
[
   "CleanSeaNet",
   "PlatFormMon"
]
```

3.6 GETALLCOUNTRIES

Retrieves all countries from the POR database

3.6.1 Protocol details

3.6.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/countries`
Description	Retrieves all the defined countries from the POR database

Method	HTTP GET
Parameters	NONE
Headers	NONE
Body	N/A
Schema	N/A

3.6.1.2 Response

Headers	NONE
Body	JSON array containing the list of countries
Schema
```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"GetAllCountries Response",
   "oneOf":[
      {
         "type":"array",
         "items":{
            "type":"object",
            "additionalProperties":false,
            "required":[
               "countryName",
               "countryCode",
               "authorityType"
            ],
            "properties":{
               "countryName":{
                  "type":"string"
               },
               "countryCode":{
                  "type":"string"
               },
               "authorityType":{
                  "type":"string"
               }
            }
         }
      },
      {
         "type":"object",
         "additionalProperties":false,
         "required":[
            "code",
            "message",
            "description"
         ],
         "properties":{
            "code":{
               "type":"integer"
            },
            "message":{
               "type":"string"
            },
            "description":{
               "type":"string"
```
 |

	} } }] }
HTTP codes	• 200 OK: request has been succesfully processed • 500 Internal Server Error: an unexpected backend error occurred

3.6.2 GetAllCountries example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/countries
```

Response:


```
[
   {
      "countryName":"Albania",
      "countryCode":"AL",
      "authorityType":"NONEUCOUNTRY"
   },
   {
      "countryName":"Austria",
      "countryCode":"AT",
      "authorityType":"MEMBERSTATE"
   },
   {
      "countryName":"Belgium",
      "countryCode":"BE",
      "authorityType":"MEMBERSTATE"
   },
   {
      "countryName":"Black Sea Commission",
      "countryCode":"XB",
      "authorityType":"REGIONALAGREEMENT"
   },
   {
      "countryName":"Bonn Agreement",
      "countryCode":"XA",
      "authorityType":"REGIONALAGREEMENT"
   },
   {
      "countryName":"Bulgaria",
      "countryCode":"BG",
      "authorityType":"MEMBERSTATE"
   },
   {
      "countryName":"EMSA Contractor",
      "countryCode":"XI",
      "authorityType":"COMPANY"
   },
```

```
    {
        "countryName":"EMSA users",
        "countryCode":"XE",
        "authorityType":"INSTITUTION"
    },
    {
        "countryName":"European Union",
        "countryCode":"EU",
        "authorityType":"INSTITUTION"
    }
]
```


3.7 GETUSERSLIST

Is an interface that provides the caller with the list of available users defined into the Planning component.

3.7.1 Protocol details

3.7.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/users`
Description	Retrieves a list of defined users
Method	HTTP GET
Parameters	N/A
Headers	NONE
Body	N/A
Schema	N/A

3.7.1.2 Response

Headers	NONE
Body	JSON array filled with the list of available users

Schema	{
	"$schema":"http://json-schema.org/draft-04/schema#",
	"title":"GetUsersList Response",
	"oneOf":[
	{
	"type":"array",
	"items":{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"id",
	"userAccount",
	"oimId"
	],
	"properties":{
	"id":{
	"type":"integer"
	},
	"userAccount":{
	"type":"string"
	},
	"oimId":{
	"type":"integer"
	}
	}
	}
	},
	{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"code",
	"message",
	"description"
	],
	"properties":{
	"code":{
	"type":"integer"
	},
	"message":{
	"type":"string"
	},
	"description":{
	"type":"string"
	}
	}
	}
	]
	}

HTTP codes	• 200 OK: request has been succesfully processed • 500 Internal Server Error: an unexpected backend error occurred

3.7.2 GetUsersList example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/users
```

Response:


```
[
   {
      "id":123,
      "userAccount":"TEST_USER1",
      "oimId":10321
   },
   {
      "id":124,
      "userAccount":"TEST_USER2",
      "oimId":10322
   },
   {
      "id":125,
      "userAccount":"TEST_USER3",
      "oimId":10323
   }
]
```


3.8 GETUSERINFORMATION

Is an interface that, given a userID it, will return the list of Operations, Organizations and the Countries the user is associated with.

3.8.1 Protocol details

3.8.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/users/[userid]`
Description	Retrieves a list of operations, organisations and countries associated with the specified userID
Method	HTTP GET
Parameters	• userID: the OIM_ID of the inspected user
Headers	NONE
Body	N/A
Schema	N/A

3.8.1.2 Response

Headers	NONE
Body	JSON object describing the detailed user info
Schema	

```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"GetUserInformation Response",
   "oneOf":[
      {
         "type":"object",
         "additionalProperties":false,
         "required":[
            "id",
            "oimId",
            "lastName",
            "firstName",
            "enabled",
            "operations",
            "organisations",
            "country",
            "roles"
         ],
         "properties":{
            "id":{
               "type":"integer"
            },
            "oimId":{
               "type":"integer"
            },
            "lastName":{
               "type":"string"
            },
            "firstName":{
               "type":"string"
            },
            "enabled":{
               "type":"boolean"
            },
            "operations":{
               "type":"array",
               "items":{
                  "type":"string"
               }
            },
            "organisations":{
               "type":"array",
               "items":{
                  "type":"string"
               }
            },
            "country":{
               "type":"string"
            },
            "roles":{
               "type":"array",
               "items":{
                  "type":"string"
               }
            }
         }
      },
      {
```

	"type":"object",
	"additionalProperties":false,
	"required":[
	"code",
	"message",
	"description"
	],
	"properties":{
	"code":{
	"type":"integer"
	},
	"message":{
	"type":"string"
	},
	"description":{
	"type":"string"
	}
	}
	}
	]
	}

HTTP codes	• 200 OK: request has been succesfully processed • 404 Not Found: user not found • 400 Bad Request: non numeric id was requested • 500 Internal Server Error: an unexpected backend error occurred

3.8.2 GetUserInformation example

Request:

```
http://<host>[:<port>]/por/rest/v1.0/users/10321
```

Response:


```
{
   "id":123,
   "oimId":10321,
   "lastName":"Name-IT-UP06-02",
   "firstName":"First-Name-IT-UP06-02",
   "enabled":true,
   "operations":[
      "CleanSeaNet"
   ],
   "organisations":[
      "Not defined (IT)"
   ],
   "country":"Italy",
   "roles":[
      "UP07:Coastal State Operational Representative",
      "UP10:Coastal State Planning Representative"
   ]
}
```

3.9 GETALLPLATFORMS

Retrieves all the configured Platforms on the EMSA Class table.

3.9.1 Protocol details

3.9.1.1 Request

Type	REST			
URL	**Error! Hyperlink reference not valid.**>			
Description	Retrieves all the configured Platforms on the EMSA Class table			
Method	HTTP GET			
Parameters	Timestamp	Long	Optional	The timestamp in milliseconds to check for the active platforms. By default the current time is used.
Headers	NONE			
Body	N/A			
Schema	N/A			

3.9.1.2 Response

Headers	NONE
Body	JSON object returning the list of sensor modes supported by the sensor with given ID
Schema	
HTTP codes	• 200 OK: request has been succesfully processed • 400 Bad Request: invalid timestamp • 500 Internal Server Error: an unexpected backend error occurred

3.9.2 GetAllPlatforms example

Request:

```
http://<host>[:<port>]/csndc-finsys-ws/rest/platforms
```

Response:

```
["GEOEYE-1","RADARSAT-2"]
```

3.10 GETALLSENSORMODES

Gets the sensor modes and respective polarizations configured for a specific platform.

3.10.1 Protocol details

3.10.1.1 Request

Type	REST			
URL Endpoint	http://{host}:{port}/csndc-finsys-ws/rest/sensorModes/<platform>&timestamp=<timestamp>			
Description	Gets the sensor modes and respective polarizations configured for a specific platform			
Method	HTTP GET			
Parameters	Timestamp	Long	Optional	The timestamp in milliseconds to check for the active platforms. By default the current time is used.
Headers	NONE			
Body **Schema**	N/A N/A			

3.10.1.2 Response

Headers	NONE
Body	JSON object returning the list of sensor modes supported by the sensor with given ID
Schema	
HTTP codes	• 200 OK: request has been successfully processed • 400 Bad Request: invalid timestamp • 404 Not Found: sensor not found • 500 Internal Server Error: an unexpected backend error occurred

3.10.2 GetAllSensorModes example

Request:

```
http://<host>[:<port>]/csndc-finsys-ws/rest/sensorModes/RADARSAT-2&timestamp=1489580625464
```

Response:

```
[
```

```
  {
    "sensorMode": [
      "SCNA",
      "SCNB",
      "SCN"
    ],
    "polarization": [
      "S"
    ]
  },
  {
    "sensorMode": [
      "XF0W1",
      "XF0W",
      "XF0W",
      "XF0S"
    ],
    "polarization": [
      "S"
    ]
  }
]
```


3.11 GETSERVICEALERTS

Gets the user Alert Level of all Oil Spills available in an acquisition, by passing the acquisition identifier (Service ID) .

3.11.1 Protocol details

3.11.1.1 Request

Type	REST
URL	`http://[host]/oas/rest/v1.0/servicealerts/[serviceid]`
Description	Retrieves all Oil Spills information from an acquisition for a given Service ID
Method	HTTP GET
Parameters	• serviceid: the id of the acquisition
Headers	UserID (e.g. UserID:GB-UP09-02)
Body **Schema**	N/A N/A

3.11.1.2 Response

Headers	NONE
Body	JSON object describing the alert info

Schema	{
	"$schema":"http://json-schema.org/draft-04/schema#",
	"title":"GetServiceAlerts Response",
	"oneOf":[
	{
	"type":"array",
	"items":{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"oilspillId",
	"alertLevel"
	],
	"properties":{
	"oilspillId":{
	"type":"string"
	},
	"alertLevel":{
	"type":"string",
	"enum":[
	"Red",
	"Yellow",
	"Green"
	]
	}
	}
	}
	},
	{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"code",
	"message",
	"description"
	],
	"properties":{
	"code":{
	"type":"integer"
	},
	"message":{
	"type":"string"
	},
	"description":{
	"type":"string"
	}
	}
	}
	]
	}

HTTP codes	• 200 OK: information found • 400 Bad Request: non numeric id was requested • 404 Not Found: service not found • 500 Internal Server Error: an unexpected backend error occurred

3.11.2 GetServiceAlerts example

Request:

```
http://twls10:81/oas/rest/v1.0/servicealerts/1707030000
(header UserID:GB-UP09-02)
```

Response:


```
[
{"oilspillId":"1707030000_S1A_IW_GRDM_1SVV_20170703T061207_20170703T061539_017303_01CE30_B587_OS_10","alertLevel":"Yellow"},
{"oilspillId":"1707030000_S1A_IW_GRDM_1SVV_20170703T061207_20170703T061539_017303_01CE30_B587_OS_2","alertLevel":"Yellow"},
{"oilspillId":"1707030000_S1A_IW_GRDM_1SVV_20170703T061207_20170703T061539_017303_01CE30_B587_OS_3","alertLevel":"Yellow"},
{"oilspillId":"1707030000_S1A_IW_GRDM_1SVV_20170703T061207_20170703T061539_017303_01CE30_B587_OS_4","alertLevel":"Yellow"}
]
```


3.12 GETOILSPILLALERT

Gets the user Alert Level of an Oil Spill when passing the Oil Spill's event id (e.g.: 1705200013_RS2_20170520_160746_0076_SCWA_VV_SGF_560615_1955_15036986_OS_1)

3.12.1 Protocol details

3.12.1.1 Request

Type	REST
URL	`http://[host]/oas/rest/v1.0/oilspillalert/[oilspillid]`
Description	Retrieves the Alert Level of an Oil Spill, given its ID
Method	HTTP GET
Parameters	• oilspillid: Oil Spill's event id
Headers	UserID (e.g. UserID:GB-UP09-02)
Body **Schema**	N/A N/A

3.12.1.2 Response

Headers	NONE
Body	JSON object describing the alert info

Schema	{
	"$schema":"http://json-schema.org/draft-04/schema#",
	"title":"GetOilspillAlert Response",
	"oneOf":[
	{
	"type":"string"
	},
	{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"code",
	"message",
	"description"
	],
	"properties":{
	"code":{
	"type":"integer"
	},
	"message":{
	"type":"string"
	},
	"description":{
	"type":"string"
	}
	}
	}
	]
	}

HTTP codes	• 200 OK: information found • 400 Bad Request: not valid id was requested • 404 Not Found: service not found • 500 Internal Server Error: an unexpected backend error occurred

3.12.2 GetOilspillAlert example

Request:

```
http://twls10:81/oas/rest/v1.0/oilspillalert/1707030000_S1A_IW_GRDM_1SVV_20170703T061207_20170703T061539_017303_01CE30_B587_OS_10
(header UserID:GB-UP09-02)
```

Response:

```
"Yellow"
```

4 NOTIFICATION INTERFACES

This chapter contains a detailed description of the new interface exposed by CSN-DC, used for notifying when the EOP Lot 1 completes the ingestion process and the generated image is available.

This interface can by disabled by configuration, and in such case it will return an error (i.e. HTTP 403 Forbidden).

4.1 SETQUICKLOOKAVAILABILITY

This interface notifies the CSN-DC that a new quicklook image is available on the EOP Lot 1. Available quicklook means that further WMS requests are clear to be sent and succesfully processed by the EOP Lot 1.

4.1.1 Protocol details

4.1.1.1 Request

Type	REST
URL	`http://[host]/por/rest/v1.0/quicklooks`
Description	The POST method must be used to notify the first occurrence of a file. A status code and a message shall be provided to inform the CSN-DC system of the successfulness of the quicklook ingestion. The PUT method may be used to update an already notified file after a re-ingestion attempt.
Method	HTTP POST / PUT
Parameters	N/A
Headers	NONE
Body	In the request payload, the following items shall be specified (see request schema below): • orderId • filename • hash • status • message

Schema	{
	"$schema":"http://json-schema.org/draft-04/schema#",
	"title":"SetQuicklookAvailability Request",
	"type":"object",
	"additionalProperties":false,
	"required":[
	"orderId",
	"fileName",
	"hash",
	"status",
	"message"
	],
	"properties":{
	"orderId":{
	"type":"integer"
	},
	"fileName":{
	"type":"string"
	},
	"hash":{
	"type":"string",
	"pattern":"^[0-9a-fA-F]{32}$"
	},
	"status":{
	"type":"string",
	"enum":[
	"OK",
	"ERROR"
	]
	},
	"message":{
	"type":"string"
	}
	}
	}

4.1.1.2 Response

Headers	NONE
Body	JSON object with the identification key of the created notification

Schema
```
{
   "$schema":"http://json-schema.org/draft-04/schema#",
   "title":"SetQuicklookAvailability Response",
   "oneOf":[
      {
         "type":"object",
         "additionalProperties":false,
         "required":[
            "orderId",
            "fileName",
            "hash"
         ],
         "properties":{
            "orderId":{
               "type":"integer"
            },
            "fileName":{
               "type":"string"
            },
            "hash":{
               "type":"string",
               "pattern":"^[0-9a-fA-F]{32}$"
            }
         }
      },
      {
         "type":"object",
         "additionalProperties":false,
         "required":[
            "code",
            "message",
            "description"
         ],
         "properties":{
            "code":{
               "type":"integer"
            },
            "message":{
               "type":"string"
            },
            "description":{
               "type":"string"
            }
         }
      }
   ]
}
```
 |
| **HTTP codes** | • 201 Created: file info succesfully notified
• **403 Forbidden**: service is disabled by configuration
• **404 - Not Found:** file info for the file not yet notified **(HTTP PUT)**
• **409 - Conflict**: file info for the file already notified **(HTTP POST)**
• 500 Internal Server Error: an unexpected backend error occurred |

4.1.2 SetQuicklookAvailability example

Request:


```
{
   "orderId":17760,

"fileName":"17760_ASA_WSM_1PXCLS20120123_220611_00000159X000_00000_51782_5498_PV.t
if",
   "hash":"efb9fb8982f33a5e4dd1060e646e94c5",
   "status":"ERROR",
   "message":"Cannot process quicklook image: not enough disk space available"
}
```


Response:


```
{
   "orderId":17760,

"fileName":"17760_ASA_WSM_1PXCLS20120123_220611_00000159X000_00000_51782_5498_PV.
tif",
   "hash":"efb9fb8982f33a5e4dd1060e646e94c5"
}
```

5 GIS VIEWER INTERFACES AND CONFIGURATION

5.1 DESCRIPTION

In the current CSN, the GIS Viewer will provide configuration options in order to integrate EOP Lot 1 with the following two functionalities:

1. The GISViewer should consume the WMS from EOP Lot1 to display images.
2. The GISViewer functionality of downloading images should retrieve the information from EOP Lot1.

These functionalities will impact respectively the following two services:

1. WMS, in order to visualize PV/PT images on GIS viewer
2. WCS, for downloading the PV/PT images

5.1.1 CSN-EOP Lot 1 Integration requirements

In order to guarantee the correct integration between CSN and EOP Lot 1, it is required that the **naming convention for the layers** will be respected. An example of layer name is: `010044:1004332_RS2_20151001_161250_0144_SCWA_VV_SCW_40704`

It it also necessary that the WMS will guarantee to serve images in the **various SRS required**.

5.1.2 Configuration

On the CSN-DC, the configuration of the server's URL for WCS and WMS is written in the `acs_global_config_emsa/emsa/*.ini` file (there is one file for each environment).

The values to be changed are the following:

```
; Address of WCS Service  (i.e. http://<host>:<port>/geoserver/wcs
WCS_SERVER = http://csndc-geoserver:7022/geoserver/wcs

; Address of WMS Service  (i.e. http://<host>:<port>/geoserver/wcs
CUSTOM_WMS_SERVER_URL =  http://twls14/geoserver/wms
BL_CUSTOM_WMS_SERVER_URL =  http://csndc-geoserver:7022/geoserver/wms
```

Please note that the URL defined by **CUSTOM_WMS_SERVER_URL** must be reachable from the user's client.

5.2 WMS SERVICE

The GIS Viewer makes two types of request to the WMS service:

- GetMap
- GetLegendGraphic

5.2.1 WMS GetMap

From GIS Viewer WMS getMap (see OGC standard for version 1.1.1)

5.2.1.1 Sample request

```
<WMSSERVERNAME>?REQUEST=GetMap&VERSION=1.1.1&EXCEPTIONS=text/xml&LAYERS=<LAYERNAME>&SERVICE=WMS&WIDTH=781&HEIGHT=322&BBOX=<BBOX>&SRS=EPSG:3395&STYLES=&FORMAT=image/png&TRANSPARENT=TRUE
```

5.2.1.2 Expected response

PNG image satisfing the request as per WMS standard

5.2.2 WMS GetLegendGraphic

from GIS Viewer WMS GetLegendGraphic (see OGC standard for version 1.1.1)

5.2.2.1 Sample request

```
<WMSSERVERNAME>?request=GetLegendGraphic&format=image/png&width=10&height=10&layer=<LAYERNAME>&style=
```

5.2.2.2 Expected Response

PNG image satisfying the request as per WMS standard

5.3 WCS SERVICE

The GIS Viewer makes two types of request to the WCS service:

- DescribeCoverage
- GetCoverage

5.3.1 DescribeCoverage

WCS describe coverage (see OGC standard for version 1.0.0)

5.3.1.1 Sample request

```
<WCSSERVERNAME>?SERVICE=WCS&VERSION=1.0.0&REQUEST=DescribeCoverage&cov
erage=<LAYERNAME>
```

5.3.1.2 Expected response

XML satisfying the request as per WCS standard

5.3.2 GetCoverage

WCS get coverage (see OGC standard for version 1.0.0)

5.3.2.1 Sample request

```
<WCSSERVERNAME>?service=WCS&version=1.0.0&request=GetCoverage&coverage
=<LAYERNAME>&bbox=<BBOX>&width=<WIDTH>&height=<HEIGHT>&CRS=EPSG:4326&f
ormat=geotiff
```

5.3.2.2 Expected response

GeoTIFF image satisfying the request as per WCS standard

5.4 GIS VIEWER DOWNLOAD PRODUCT

Retrieves the relative URL to allow the user's client to download the image product.

When the client will call the returned URL, the EOP Lot1 image server shall return a zipped file containing the TIFF image, with the requested resolution, together with the license file. The server shall also provide the necessary HTTP headers to correctly identify the zip filename for the client.

5.4.1 Protocol details

5.4.1.1 Request

Type	REST
URL	`http://[host]/eodc/downloadProduct/[serviceIdentifier]/[resolution]`
Description	Retrieves the client-side relative url to download the zipped product image
Method	HTTP GET
Parameters	• serviceIdentifier: the ID of the requested product • resolution: enumerated value, "high" or "low", default as "low"
Headers	NONE
Body	N/A
Schema	N/A

5.4.1.2 Response

Headers	NONE
Body	JSON object describing the download product url

Schema	{
	"$schema":"http://json-schema.org/draft-04/schema#",
	"title":"DownloadProduct Response",
	"oneOf":[
	{
	"type":"string",
	},
	{
	"type":"object",
	"additionalProperties":false,
	"required":[
	"code",
	"message",
	"description"
	],
	"properties":{
	"code":{
	"type":"integer"

	<pre>},
	"message":{
	"type":"string"
	},
	"description":{
	"type":"string"
	}
	}
	}
	]
	}</pre>

HTTP codes	• 200 OK: service identifier found • 404 Not Found: service not found • 500 Internal Server Error: an unexpected backend error occurred

5.4.2 DownloadProduct example

Request:

```
http://<host>[:<port>]/eodc/downloadProduct/1703280021_17MAR28091138-S2AS-1703280021-RGB_WV2RGB/high
```

Response:

```
"/eodc/eop/2419b1b8942921de447afdb095a2bea9"
```

or, in case of error:

```
{
  "code": 1234,
  "message": "wrong serviceId",
  "description": "unknon product identifier"
}
```

- END OF DOCUMENT -