

EMSA CleanSeaNet
Data Centre [CSNDC]
Technical Design Document (TDD)

Insert reference or version 7.1
Date: 29/05/2015

TABLE OF CONTENTS

1	Introduction.....	13
1.1	Purpose and Objectives	13
1.2	Reference documents	13
1.3	Abbreviations and acronyms	14
2	Infrastructure and Environments.....	18
2.1	General Guidelines.....	18
2.2	Availability requirements	18
2.2.1	High Availability/Load Balancing Cluster methods	18
2.2.2	Availability/DR	18
2.3	Middleware	19
2.4	Environments & SW Versions management.....	19
2.5	Environment	20
2.6	Component scalability	20
2.7	Production environment	21
2.8	Pre-Production Environment	21
2.9	TEST Environment.....	22
2.10	Data Budget	22
2.11	Backup	23
3	Architectural Decisions	24
3.1	Architectural design approach.....	24
3.2	Existing components (COTS).....	24
3.3	Components developed/adapted	25
3.4	Internal interfaces communication mechanism	26
3.5	Introduction to PDS (Payload Data Segment)	26
3.5.1	Data Driven	27
3.5.2	Modularity.....	30
3.5.3	Scalability	30
4	Logical View.....	32
4.1	System Overview	32
4.2	DAM – Data Management.....	33
4.2.1	DAM Overview	33
4.2.2	DAM Decomposition	34
4.2.2.1	Data Store	34
4.2.2.2	Mass Storage	35
4.2.2.3	Webcat.....	35
4.2.2.4	WebCat Feeder.....	37
4.2.3	Geoserver logical architecture	38
4.2.4	Implementation of the Catalogue Services	39

4.2.5	Specific configuration of Deegree to implement the ebRIM scheme and the EOP profile	41
4.2.6	Implementation of the ingestion of raster data	41
4.3	IIF – Ingestion and Interfaces	42
4.3.1	IIF Overview	42
4.3.1.1	Integrity, Format and Quality control checks	43
4.3.1.2	Auxiliary files Timeliness Verification	43
4.3.2	IIF Decomposition	44
4.4	UMA	48
4.4.1	UMA Overview	48
4.4.2	List of CSNDC users and roles	51
4.4.3	Implementation of the Data Access Policy (DAP) Management	53
4.4.4	Implementation of the Operation-Based Data Access	55
4.5	PMA – Process Management	57
4.5.1	PMA Overview	57
4.5.2	PDS Thin Layer	58
4.5.3	KEO	58
4.5.3.1	KAFE Servers	59
4.5.3.2	Feature Extraction Processor	59
4.5.3.3	Data-Flow Explained	60
4.5.3.4	Architectural Overview	60
4.5.3.5	FEP Engine	61
4.5.3.6	FEP Actuator	63
4.5.3.7	FEP Module Ingestion	63
4.5.3.8	FEP Designer	65
4.5.3.9	The PROCESSOR calling mechanism	69
4.5.3.10	Configurability and usage of the Thin Layer inside the CSN-DC PMA	73
4.6	WUP – Web User Portal	74
4.6.1	WCMS	79
4.6.2	Plugin Container	79
4.6.3	WebGIS application	79
4.6.4	Oil spill prediction models integration from the WUP	84
4.7	PDE – Product Delivery	85
4.7.1	PDE Overview	85
4.7.2	PDE Decomposition	85
4.7.3	Details on the alerting procedure	90
4.8	POR – Planning and Ordering	93
4.8.1	POR Overview	93
4.8.2	POR Main tasking phases	93
4.8.2.1	POR Planning phase	94
4.8.2.2	POR Tasking allocation phase	94
4.8.3	POR Approval phase	96
4.8.4	Usage of Service Types	97
4.8.5	POR decomposition	100
4.9	JOU - Journaling	103
4.9.1	Client Layer	104

4.9.1.1	Web Client	104
4.9.2	Presentation Layer	104
4.9.2.1	Portlet.....	104
4.9.2.1.1	Access control	104
4.9.2.1.2	Navigation	105
4.9.3	Application Layer.....	105
4.9.3.1	Core Business Functions	105
4.9.3.1.1	Journaling data acquisition.....	105
4.9.3.1.2	Journaling data preparation	105
4.9.3.1.3	Journaling data storage	105
4.9.3.1.4	Journaling data update	106
4.9.3.1.5	Audit information storage	106
4.9.3.1.6	Image delay calculation	106
4.9.3.1.7	Force frame delivery time and quality	106
4.9.3.1.8	Store documents sent to SPs	106
4.9.3.1.9	Store references to products sent to users.....	107
4.9.3.1.10	Store alerts/reports sent to coastal states	107
4.9.3.2	Reporting	107
4.9.3.2.1	Jasper Reports	107
4.9.3.2.2	Get/Add product category.....	108
4.9.3.2.3	Report contest	108
4.9.3.2.4	Mark report as final	108
4.9.3.3	Logging	109
4.9.3.3.1	Change log level at runtime	109
4.9.4	Integration Layer	109
4.9.4.1	Integration Infrastructure.....	109
4.9.4.1.1	External Services.....	109
4.9.5	Data Access Layer	109
4.9.5.1	Database Management.....	109
4.9.5.1.1	Database access	109
4.9.6	Data Layer.....	110
4.9.6.1	Journaling	110
4.10	Communication Mechanisms (COM)	110
4.10.1	Client Layer	111
4.10.1.1	Web Client	112
4.10.2	Presentation Layer	112
4.10.2.1	Portlet.....	112
4.10.2.1.1	Access control	112
4.10.2.1.2	Navigation	112
4.10.3	Application Layer.....	113
4.10.3.1	Document Library.....	113
4.10.3.1.1	Document management.....	113
4.10.3.1.2	Search	113
4.10.3.2	Forum.....	113
4.10.3.2.1	Forum content management	113
4.10.3.2.2	Search	113
4.10.3.3	WIKI	114

4.10.3.3.1	WIKI content management	114
4.10.3.3.2	Search	114
4.10.3.4	Calendar	114
4.10.3.4.1	Event management	114
4.10.3.4.2	Search	114
4.10.4	Data Access Layer	115
4.10.4.1	Database Management.....	115
4.10.4.1.1	Database access.....	115
4.10.5	Data Layer.....	115
4.10.5.1	Document Library.....	115
4.10.5.2	Forum.....	115
4.10.5.3	WIKI	115
4.10.5.4	Calendar	116
4.11	FinSys – Financial System	117
4.11.1	Client Layer	117
4.11.1.1	Web Client	118
4.11.2	Presentation Layer	118
4.11.2.1	Portlet.....	118
4.11.2.1.1	Access control	118
4.11.2.1.2	Navigation	118
4.11.3	Application Layer.....	119
4.11.3.1	Core Business Functions	119
4.11.3.1.1	Frame Price Calculation	119
4.11.3.1.2	Task Form Generation	119
4.11.3.2	Reporting	119
4.11.3.2.1	Jasper Reports	119
4.11.3.2.2	Generate Financial Report.....	120
4.11.3.3	Logging	120
4.11.3.3.1	Change log level at runtime	120
4.11.4	Integration Layer	121
4.11.4.1	Integration Infrastructure.....	121
4.11.4.1.1	External Services	121
4.11.5	Data Access Layer	121
4.11.5.1	Database Management.....	121
4.11.5.1.1	Database access.....	121
4.11.6	Data Layer.....	121
4.11.6.1	Financial System.....	122
5	Technical Implementation View	123
5.1	DAM - Data Access Management.....	123
5.2	IIF - Ingestion and interfaces.....	126
5.2.1	Ingestion of AIS data	130
5.2.2	Notification service	133
5.3	UMA - User Management	137
5.3.1	Technical implementation of the Data Access Policy enforcement	139
5.3.2	Technical Implementation of Operation Based access control.....	142

5.4	PMA - Process Management	146
5.5	WUP - WEB USER PORTAL	149
5.6	PDE - Product Delivery	150
5.6.1	Component diagram and main interfaces with other components	150
5.6.2	Dynamic behaviour	151
5.7	POR Planning and ordering	155
5.7.1	Component diagram and main interfaces with other components	155
5.7.2	Dynamic behaviour	156
5.7.3	Description of the message exchanges between POR and JOU/FinSys	161
5.8	JOU	170
5.9	COM	172
5.10	Oil Spill Models integration and workflow	175
5.11	Archive and Restore Tools	178
5.11.1	Overview of the functionality	178
5.11.2	Implementation logic	179
5.11.2.1	Determine data to be archived	182
5.11.2.2	Moving physical files from online to archive location	183
5.11.2.3	Moving AIS data from the online table into the offline table	184
5.11.2.4	Removing Geoserver layers	184
5.11.2.5	Setting the status to archived	184
5.11.2.6	Determining data to be restored	184
5.11.2.7	Copying physical files back from the archive location into the online location	185
5.11.2.8	Moving AIS data from the offline table into the online table	185
5.11.2.9	Setting the status to delivered	185
5.11.3	Implementation details and handling of non-nominal cases	185
5.12	Implementation of the Business Continuity Facility (BCF) support function	188
5.12.1	Start/stop scripts	189
5.12.2	Configuration scripts	189
5.12.3	Example of switch over between EMSA PROD and BCF	190
6	Deployment View	191
6.1	First decomposition: tiers	191
6.2	Portal Tier	191
6.3	Application Tier	192
6.4	Database Server Tier	194
6.5	Identity management TIER	195
7	Implementation view	196
8	Data View	196
8.1	DAM	196
8.1.1	EOP metadata	196
8.1.2	Oil Spill feature and Detected Ship feature	198
8.2	POR	213
8.3	COM & JOU	214
8.4	FinSys	217
9	External Interfaces	220

10 Internal Interfaces.....	221
11 Annex -A - Security Concepts.....	226
11.1 Clean Sea Net Security Concepts.....	226
11.1.1 High level security objectives.....	226
11.1.2 CSN Software Development Life Cycle	226
11.1.3 Manual & Automated Verification.....	227
11.1.3.1 Dynamic Analysis.....	227
11.1.3.1.1 Vulnerability Scanning	227
11.1.3.1.2 Penetration Testing	229
11.1.3.2 Static Analysis (Source Code Scanning)	1
11.1.4 External Entities to CSN Threats Analysis.....	2
11.1.5 About Logging.....	4
11.2 Security related to Apache reconfigurations	4
11.3 Security related to reflected content and reflected cross-site scripting.....	5
12 Annex B - Alerting Parameters	6
13 Annex C - Example of POR Planning files	10
14 Annex D – Example of XML file for triggering the Oil Spill Model WPS	25

TABLE OF FIGURES

Figure 2-1 – Middleware view	19
Figure 2-2 CSNDC deployment diagram	20
Figure 3-1 Data Driven in the Production System: the Thin Layer	29
Figure 4-1 - DAM high level architecture	33
Figure 4-2 - DAM decomposition	34
Figure 4-3 – Geoserver deployment	39
Figure 4-4 - EO Catalogue specifications relationships	40
Figure 4-5 – IIF as the CSN-DC file-based data gate	42
Figure 4-6 – IIF Decomposition	44
Figure 4-7 – FTP with redundant servers (export/dissemination case)	47
Figure 4-8 Overview of the User Management in CSNDC	48
Figure 4-9: FEP Engine class diagram	61
Figure 4-10: KAFE Engine/Actuators architecture.	62
Figure 4-11: FEP actuators architecture.....	64
Figure 4-12: FEP Module Manager dialog.....	65
Figure 4-13: FEP designer window.	66
Figure 4-14: Processing module shapes.	67
Figure 4-15: Anatomy of a processing module.....	67
Figure 4-16 - Thin Layer architecture on a distributed environment.....	68
Figure 4-17 Thin Layer activation mechanism.....	71
Figure 4-18 WUP context and main interfaces.....	74
Figure 4-19 WUP as a DAM's services client.....	75
Figure 4-20 high level view of the data model accessed by the WUP	76
4-21 - WUP decomposition	79

Figure 4-22 SIBILLA based channels	80
Figure 4-23 Rich flex client init sequence	80
Figure 4-24 RichFlex client decomposition	81
Figure 4-25 Standard javascript client init sequence	82
Figure 4-26 Library hierarchy for the standard javascript client	82
Figure 4-27 – PDE Decomposition	85
Figure 4-28 Standing order flow	86
Figure 4-29 - Report generation workflow	89
Figure 4-30 - Alert Management Component Diagram	90
Figure 4-31 - Dynamic model of the alerting process	91
Figure 4-32 - POR Tasking phase	95
Figure 4-33 Flow chart illustrating the approval phase	96
Figure 4-34 – UI for defining service types	99
Figure 4-35 – Dialog window for defining service type for a given cart	100
Figure 4-36 - POR first level of decomposition	101
Figure 4-37: Journaling Logical View	103
Figure 4-38: Communication Mechanisms Logical View	111
Figure 4-39: Journaling Logical View	117
Figure 5-1 - DAM main high level ingestion scenario	123
Figure 5-2 - WUP <-> DAM sequence diagram	124
Figure 5-3 - Info collection for Alert creation sequence	125
Figure 5-4 Mechanism for receiving the data packages from the SPs	127
Figure 5-5 - Ingestion chain workflow	128
Figure 5-6 – Component diagram focussing on the dependencies between IIF, JOU, POR and FINSYS	129
Figure 5-7 - Workflow for managing ingestion of a EOP (and QNO)	129

Figure 5-8 – WebCatFeeder interfaces with COTS.....	130
Figure 5-9 – Vessel Traffic information retrieval sequence diagram	133
Figure 5-10 – Notification Service Component Diagram	135
Figure 5-11 – Notification Service sequence diagram.....	136
Figure 5-12 - User Management Component Diagram	137
Figure 5-13 - Interaction diagram for the user management.....	138
Figure 5-14 High level architecture of the Data Access components	139
Figure 5-15 – Simplified user model focussed on the user-operations relationships.....	142
Figure 5-16 Simplified model for service ID to operations relationships	145
Figure 5-17 - Ingestion processing and display of Service Provider data	146
Figure 5-18 - Complex chain including execution of an external hosted process	147
Figure 5-19 PMA processing and request for STIRES data.....	148
Figure 5-20 – Component diagram for Alerting and its main interfaces.....	150
Figure 5-21 - Management of Subscriptions	151
Figure 5-22 Systematic distribution of data	152
Figure 5-23 Generation of reports and alerts	154
Figure 5-24 POR JOU FINSYS and GIS Viewer component diagram	155
Figure 5-25 – Setting the default coastal state requirements using the POR	156
Figure 5-26 – CSN-SD creates the list of scenes allocated by Coastal States in the various formats.	157
Figure 5-27 – CSN-SD sends EOLI order and update order status	158
Figure 5-28 POR ordering data (tools different from EOLI).....	158
Figure 5-29 Finalisation of an order from the POR	159
Figure 5-30 - POR approval sequence.....	161
Figure 5-31 Information exchange between POR and JOU/FinSys.....	162
</TaskMessage>Figure 5-32 Example for the message type <i>TaskMessage</i> , sent when the PO starts a tasking.....	164

Figure 5-33 Example of message sent when a scene is allocated, SceneMessage	164
Figure 5-34 Example for the message type <i>TaskMessage</i> , sent when the PO starts approval	165
Figure 5-35 Example of <i>TaskInfoList</i> , sent when the PO starts approval	166
Figure 5-36 Message of type TaskInfoList, example sent when the FO approves an order	166
Figure 5-37 Example of ReportInfo message generated when an order is approved.....	167
Figure 5-38 Example of NewOrder message generated when a tasking is completed (approved by AO)	169
Figure 5-39 - Dynamic Flow of Journaling information process	170
Figure 5-40 - Dynamic Flow of class request	170
Figure 5-41 - Dynamic Flow of force class request	171
Figure 5-42 - Dynamic Flow of create Forum content	172
Figure 5-43 - Dynamic Flow of create Forum content from an EO product	172
Figure 5-44 - Dynamic Flow of delete WIKI content	173
Figure 5-45 - Dynamic Flow of search documents	173
Figure 5-46 - Dynamic Flow of post calendar event	173
Figure 5-47 – Component diagram illustrating the elements involved in the Oil Spill model processing..	176
Figure 5-48 – Oil spill modelling execution workflow.....	177
Figure 5-49 – Archive / restore functionality component diagram	180
Figure 5-50 – Archiving process	181
Figure 5-51 – Restoring process	182
Figure 8-1 ER diagram for EOP metadata	197
Figure 8-2 Oil Spill feature data model	199
Figure 8-3 Detected Ship feature data model.....	208
Figure 8-4 ER diagram for detected ship and oil spill metadata.....	211
Figure 8-5 ER for the POR component	213
Figure 8-6: COM and JOU data Tables	214

1 Introduction

1.1 Purpose and Objectives

This document constitutes the technical design documentation related with the CleanSeaNet Data Centre Information System.

1.2 Reference documents

Document Title	Identifier	Internal Reference
Invitation to Tender concerning the development of “EMSA CleanSeaNet Data Centre”	EMSA/OP/06/2009	[ITT]
Tender Specifications	Enclosure I	[E-I]
Draft Framework Contract	Enclosure II	[E-II]
Price Grid	Enclosure III	[E-III]
Compliance Matrix	Enclosure IV	[E-IV]
ICT Architecture, System and application Technical Landscape, version 20 from 06/02/2013.	Enclosure V	[E-V]
Project delivery	Enclosure VI	[E-VI]
Working procedures and service requirements	Enclosure VII	[E-VII]
Ordering Service for Earth Observation Products, version 0.9.4, date: 2008-09-05	OGC 06-141r2	[HMA-ORD]
OGC Catalogue Services Specification 2.0 Extension Package for ebRIM Application Profile, version 0.2.2, date: 2008-10-23	OGC 06-131r5	[HMA-CAT]
OGC OpenGIS Sensor Planning Service Application Profile for EO Sensors, version 0.9.5, 19/11/2007	OGC 07-018	[HMA-SPS]
Common Alerting Protocol http://www.oasis-open.org/committees/download.php/15135/emergency-CAPv1.1-Corrected_DOM.pdf http://www.oasis-open.org/apps/org/workgroup/emergency/download.php/14205/emergency-CAPv1.1-Committee%20Specification.doc	CAP-V1.1	[CAP]
OGC User Management Interfaces for Earth Observation Services, version 0.0.4, 30/06/2009	OGC 07-118r1	[HMA-IDM]
Secure Hash Standards (SHA-1) National Institute of Standards and Technology http://csrc.nist.gov/cryptval/shs.htm		[SHA-1]
ACS Quality Guidelines for HMI Design, issue 3.2 17/07/2009	SW-PA-ACS-QA-0103	[HMI-GL]
Mission Capacity Planning User Guide, 1.0 30/06/2009	MCP-MA-ACS-GMES-0116	[MCP-UG]

Document Title	Identifier	Internal Reference
Certified email description	http://www.certifiedemail.net/what-is-certified-email.php	[CE-EMAIL]
PDS Generic Processor ICD	CSNDC-ID-ACS-EMSA-0103	[PDS-ICD]
CSN-DC Functional Design Document, Issue 2.0 08/01/2010	CSNDC-DD-ACS-EMSA-0101	[FDD]
CSN-DC External Interface Control Document, Issue 1.3.4-3, 08/04/2015	CSNDC-ID-ACS-EMSA-0104	[EXT-IF-ICD]
EMSA ICT Architecture System and Application Technical Landscape, issue 14, 17/11/2009		[ICT-SATL]
Uncover security design flaws using the STRIDE approach	http://msdn.microsoft.com/en-us/magazine/cc163519.aspx	[STRIDE]
CSN-DC Installation Manual, Issue 1.0	CSNDC-IN-ACS-EMSA-108	[INS]
CSN-DC Operations and Maintenance Manual, Issue 2.8	CSNDC-OMM-ACS-EMSA_0107	[OMM]
OIM - CSN Target System Webservice Specification, Issue 2.5		[OIM-CSN]
Alerting Template Detailed Description, Anbex 2 to CSN DC v1.2 EMSA Priorities list.		[ALR]

1.3 Abbreviations and acronyms

Table 1-1 - Abbreviations and Acronyms

Abbreviation	Definition
ACS	Advanced Computer Systems A.C.S. S.p.A.
AIP	Algorithmic Image Processing
AIS	Automatic Identification System for vessels
AJAX	Asynchronous JavaScript and XML
AOI	Area Of Interest
AOI	Area of Interest
API BPEL	Application Programming Interface Business Process Execution
CAP	Common Alerting Protocol
	Language Common Alerting Protocol Coordinated Data access
CAP CDS CIM COM	System Cataloguing of ISO Metadata Communication Component
COTS	Commercial off-the-shelf
CS	All EU Member States, Candidate countries and EFTA Coastal States
CSA	Canadian Space Agency
CSD	CSN service desk (planning and ordering)
CSN	CleanSeaNet
CSN-DC	CleanSeaNet Data Centre

Abbreviation	Definition
CSS	Cascading Style Sheets
CSW	Catalogue Services for the Web
DAIL	Data Access and Integration Layer
DAM	Data Management Component
DBA	Database Administrator
DC	Data Centre
DDS EC EGEMP	Data Dissemination System European Community Expert Group of Experts on satellite based Monitoring of sea-based oil Pollution
DGF	Data Gate Facility
DREAM	Decision Support and Real Time EO Data Management
EDA	Event Driven Architecture
EFTA	European Free Trade Association (Iceland, Norway, Switzerland, and Liechtenstein)
EIS	European Index Server
EMSA	European Maritime Safety Agency
EMSA ENC	European Maritime Safety Agency Electronic Nautical Charts
ENC	Electronic Nautical Charts
EO	Earth Observation
ESAPI	Enterprise Security API
ERS	European Remote Sensing satellite
ESA	European Space Agency
ESB	Enterprise Service Bus
EU	European Union
EU MS	European Member States
EUSC	European Union Satellite Centre
FEP	Feature Extraction Processor
FTP	File Transfer Protocol
GCM	GMES Contributing Missions
GMES	Global Monitoring of Environment and Security
GML	Geographic Markup Language
GMLJP2 GSCDA	GML for JPEG2000 Imagery GMES Space Component Data Access
GUI HMA HTML HTTP ICD	Graphical User Interface Heterogeneous Mission Accessibility
HTTPS	Hyper Text Mark-up Language Hyper Text Transfer Protocol
ICT	Interface Control Document
IMDatE	Hypertext Transfer Protocol over Secure Socket Layer
IMO	Information Communication & Technology
IPF	Integrated Maritime Data Environment
ISM	International Maritime Organisation
J2EE	Instrument Processing Facility
JOU	Intelligent Storage Management
JPEG	Java 2 Enterprise Edition
JSP	JOUrning System
KML	Joint Photographic Experts Group
KMZ	Java Server Pages
LRIT	Keyhole Mark-up Language
	Keyhole Mark-up Language zipped
	Long Range Identification and Tracking

Abbreviation	Definition
MAT	MultiMission Planning Analysis Tool
MCP	Mission Capacity Planning
MDA	MacDonald Dettwiler and Associates
	Operator Member States Multi-mission User Interface Service Organisation for the
MS MUIS OASIS	Advancement of Structured Information Standards
MSSO	Maritime Support Service
NRT	Near Real Time
OGC	Open Geospatial Consortium
OGC PDE	Open GeoSpatial Consortium Product Delivery component
OTS	Off The Shelf Software (used commonly to point to reused SW)
PDS	Payload Data Segment
PKI	Public Key Infrastructure
PKI	Public Key Infrastructure
PMA POR PQP QoS	Process Manager component Planning and Ordering component
	Project Quality Plan Quality of Service
QoD	Quality of Data
RBAC	Role Based Access Control
REST	Representational Transfer State
RSS	Really Simple Syndication
SAML	Security Assertion Markup Language 1.1, OASIS http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security
SAR	Synthetic Aperture Radar
SDLC	Software Development Life Cycle
SDO	Service Desk Operator
SLA SO	Service Level Agreement CSN Satellite owner
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SP	CSN Service Provider
SSL SSN	Secure Sockets Layer SafeSeaNet
SST STIRES SUM	Sea Surface Temperature SafeSeaNet Tracking Information Relay
	Exchange System Software User Manual
SYS SVG	System Control component Scalable Vector Graphics
THETIS	The Hybrid European Targeting and Inspection System
TLS	Transport Layer Security
TOI	Time Of Interest
UDDI	Universal Description, Discovery and Integration Service
URL	Uniform Resource Locator
USNG	User Service Next Generation
W3C	World Wide Web Consortium
WBS	Work Breakdown Structure
WCS	Web Coverage Service
WFM	Work Flow Manager
WFS	Web Feature Service
WLI	Web Logic Integrator
WML WMS WSDL	Wireless Markup Language Web Mapping Service Web Services
	Definition Language
WMS	Web Map Service

Abbreviation	Definition
WPS	Web Processing Service
XML WUP	eXtensible Markup Language Web User Portal Component

2 Infrastructure and Environments

This chapter describes the system infrastructure, from the Hardware and COTS point of view.

2.1 General Guidelines

The CSN-DC will be based on a virtualised environment, following the Technical Landscape guidelines [E-V], with the exception of the Oracle production database. A virtualised environment greatly simplifies the application deployment, augments the server availability and reduces the physical needed resources.

During the various releases, an analysis of the workload of the single machines will be carried out in order to correctly size the characteristics of physical and virtual machines.

2.2 Availability requirements

[The CSN DC shall be capable to be operated on fully redundancy equipment including local back-up servers working in switch-over or load-balancing as well on remote back-up servers with should be ready to overtake crucial activities in case of disaster]

Period	Availability	Maximum downtime
Day	97,5%	36 minutes
Month	99,5%	3 hours 36 minutes
Year	99,9%	8 hours 45 minutes

2.2.1 High Availability/Load Balancing Cluster methods

In order to achieve the desired availability requirements, the following methods are used to cluster various services:

- **Weblogic:** CNS-DC web application will be deployed in a weblogic application server configured in active/active clustering, using the weblogic clustering solution. This will provide also load balancing capabilities.
- **Oracle:** for the production environment the Oracle installation will be configured by EMSA in a RAC environment. This will provides High Availability and Load Balancing
- **VMware:** Both vSphere HA (High Availability) and Fault Tolerance options will be used. Since the Fault Tolerance option implies doubling the CPU and memory resources needed by the components, only servers containing mission critical components will be deployed in virtual machines with this option turned on. VMware HA option is acceptable for non-mission critical components.
- **Some ACS components intrinsically provide load balancing/fault tolerance mechanisms (e.g. the thin layer, see component scalability paragraph for details).**

2.2.2 Availability/DR

In order to find the right balance between the amounts of needed hardware and to ensure the desired availability, each machine is classified in one of the following categories:

Classification	SLA	Disaster Recovery	Note
VHC –Very High Critical	Yes	Yes	High Availability ensured by <ul style="list-style-type: none"> • VMware Fault Tolerance or • Weblogic clustering or • Oracle RAC or • Component deployment
HC – High Critical	Yes	Yes	High Availability ensured by VmWare HA
MLNC – Medium, low non critical	No	No	No HA (if physical), no DR

Very High Critical machines: the mission critical servers. The unavailability of one of these machines has direct impact on the service. The VHC servers are replicated in Disaster Recovery, and protected by Oracle RAC, Weblogic Cluster or VMware Fault Tolerance option.

High Critical machines: Like VHC but it is acceptable a short downtime of the server in case of hardware fault.

Medium, low or non critical: The MNLC machines are servers not hosting vital services. It is acceptable to run in disaster recovery situation without one of these services. . If the machine is physical, a downtime of hours or days is acceptable.

The estimated sizing of the machines for the production environment is reported in § 2.7.

2.3 Middleware

The following figure shows the middleware that will be used for the CSN-DC implementation.

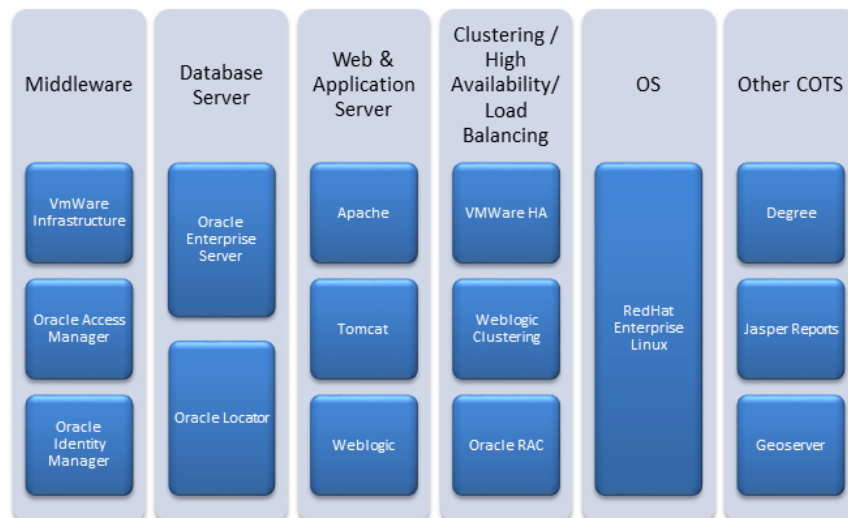


Figure 2-1 – Middleware view

Usage of the various components will be discussed on a case by case basis in the following text.

2.4 Environments & SW Versions management

Three different environments are foreseen:

- Test
- Pre Production
- Production

They are meant indeed to be used with slightly different purposes in the different project phases and have also a throughput limit with respect to the Production environment set to 50% for the Pre Production and 25% for the Test/Development one.

2.5 Environment

The physical deployment is reported in the following picture.

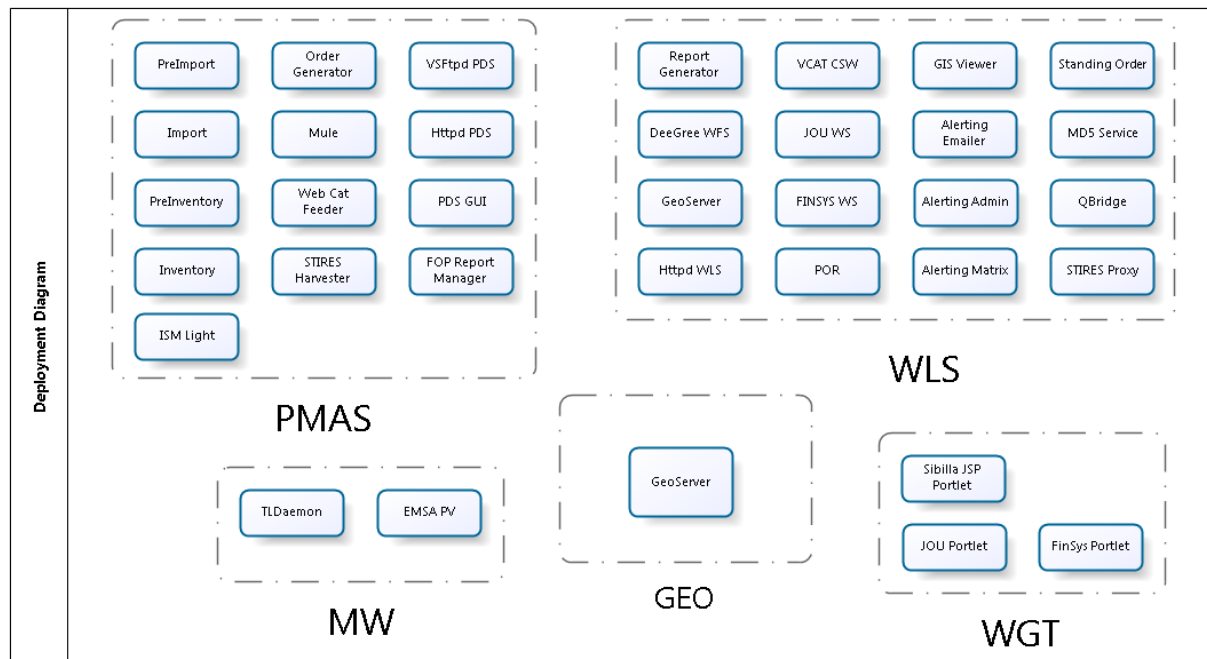


Figure 2-2 CSNDC deployment diagram

2.6 Component scalability

The Thin layer component, that is the component responsible for the SAR image processing, is currently deployed on two node on both the Production and PreProduction environments, while in the Test environment it is deployed on one node only.

Moreover, CSN-DC Oracle Database is configured in a RAC configuration.

Same consideration holds for the WebLogic cluster for the Business Layer, that is composed of 2 nodes in both Production and PreProduction environments, while it has only one node in the Test Environment.

2.7 Production environment

Name	Number of nodes	CPU	RAM	Cluster	Storage (GB)	Note
PMA-PDS-SRV	1	2 VCPU	8GB		100	
PMA-PDS-WST	2	2VCPU	4GB		100	
Portal-SRV	2	4VCPU	16GB	Weblogic cluster	100	
DAM-SRV	2	4VCPU	16GB	Weblogic Cluster	200	
DAM-MASS	2	2VCPU	8GB	VMWare FT	100 + 3,5 TB each year	
Oracle DB	2	4 VCPU	16GB		100	Oracle is hosted in physical machines

2.8 Pre-Production Environment

In pre-production environment the servers are deployed on single nodes.

Name	Number of nodes	CPU	RAM	Cluster	Storage (GB)	Note
PMA-PDS-SRV	1	2 VCPU	4GB		80	
PMA-PDS-WST	1	2 VCPU	4GB		40	The amount of RAM and CPU depends on the processes that are hosted in the thin layer. This is only a rough estimation.
Portal-SRV	2	2 VCPU	4GB	Weblogic cluster	36	Presentation Tier
DAM-SRV	2	2 VCPU	8GB	Weblogic cluster	100	Business Tier
DAM-MASS	2	2 VCPU	4GB		100	Storage for the ISM virtual file system is half of the production environment.
Oracle DB (in RAC)	2	2 VCPU	8GB		100	Since the Oracle DB is a leverage EMSA asset, sizing of this machine (number of nodes) will be analysed with the EMSA DBA. Total amount of space needed by the various instances are contained in the Data

Budget section.

2.9 TEST Environment

For the test/development environment the virtual machines characteristics are the same of the pre-production environment, but the density of VmWare virtual machines on physical hardware is double.

Name	Number of nodes	CPU	RAM	Cluster	Storage (GB)	Note
PMA-PDS-SRV	1	2 VCPU	4GB		40	
PMA-PDS-WST	1	1 VCPU	4GB		20	
Portal-SRV	1	2 VCPU	4GB		36	Presentation Tier
DAM-SRV	1	2 VCPU	4GB		100	Business Tier
DAM-MASS	1	2 VCPU	4GB		36	
Oracle DB	1	2 VCPU	4GB		36	

2.10 Data Budget

Data Type	Repository	MB/Day	GB/year
AIS data	Webcat Feature DB	14,65	5,22
SAR	ISM file system	15000,00	5475
SAR metadata	Webcat Ebrim DB	0,07	0,03
SAR Radiometric Corrected	ISM file system	750,00	267,07
Oil spill Warnings (tar)	ISM file system	15,00	5,34
Oil spill Warnings (metadata)	Webcat Feature DB	15,00	5,34
Oil Spill Notifications (tar)	ISM file system	15,00	5,34
Oil Spill Notifications (metadata)	Webcat Feature DB	15,00	5,34
SAR wind and wave/swell (tar)	ISM file system	0,60	0,21
SAR wind and wave/swell (metadata)	Webcat Feature DB	0,60	0,21
SAR Vessel Detection (tar)	ISM file system	0,15	0,05
SAR Vessel Detection (metadata)	Webcat Feature DB	0,15	0,05
SAR Image Quality Notifications	ISM file system	0,15	0,05
Quality Reports	ISM file system	15,00	5,34
MyOcean data	ISM file system	1.000,00	356,10
MD5 sub package data	ISM file system	0,15	0,05
External Process data (tar)	ISM file system	100,00	35,61
External Process data (metadata)	Webcat Feature DB	100,00	35,61

Region of interests	POR DB	1,00	0,36
Feasibility Planning	POR DB	0,10	0,04
Acquisition Status Files	POR DB	0,10	0,04
COM data	Liferay DB	38,00	14,00
JOU data	JOU DB	23	4,7
TOT		16374	6663

Note:

The original approximate native1 image size is:

- ASAR: 150 MB
- Radarsat 1/2 in 16 bit is 700 MB

For the estimation of SAR radiometric corrected images, the following assumptions have been made:

- An average Radarsat image is 12,000x12,000 pixel, which is higher to the ASAR size
- The output images are 8-bit, no compression
- The pyramidal resolution is doubling the space needed
- Resulting processed image size 300 MB for Radarsat.

Therefore, assuming the worst case scenario (15 Radarsat images per day):

- Radiometrically corrected data with pyramidal processing: $300 \times 15 \text{ MB} = 4,500$
- SAR-Native1 data: $700 \times 15 = 10,500$
- Total: 15,000 MB/day

For COM data, the estimation has been done with a figure of 10 MB/day of images and attachments for the discussions.

For JOU data, the estimation has been done considering 10 Kb per record, 7 records per image and 15 images per day.

NOTE: FinSys data is considered negligible, as it is not updated routinely.

2.11 Backup

The CSN-DC will be integrated in the EMSA leverage backup infrastructure. No particular requirements (dedicated agents) are needed. The detail of each machine backup policy is illustrated in the Operational and maintenance documentation.

3 Architectural Decisions

3.1 Architectural design approach

The architecture implemented for the CSN-DC matches the EMSA needs to have a system based on the state-of-the-art technologies, having a SOA architecture (where applicable), as derived from previous Projects such as the “data driven” PDS for the ESA Earth Explorer, the User Service Next Generation for ESA and the EUSC Reference Facility Project where workflows were integrated for similar applications.

It is indeed also important to mention that the proposed architecture, derived from the above main Projects, is especially suited for Near Real Time systems i.e. systems requiring a very small “latency” in the system infrastructure. This is particularly true for the PDS, which is a well consolidated ACS infrastructure implemented in many ground segments with different configurations of the same COTS.

Due to the fact that the reused SW has been designed for scientific missions, the overall SW and HW architecture is highly scalable and can be tuned according to the Operational needs.

Moreover the driving architectural choice to have facilities and components which are independent each other shall be an asset whenever EMSA would decide to add new functionalities to its existing specified requirements.

It is important to state that ACS has given **maximum preference to Open Source software** which integrated by COTS solutions (SW already used and operational in other system) provides a consistent robustness to the EMSA CSN DC System.

3.2 Existing components (COTS)

The following list shows the COTS that are used for the CSN-DC, and their mapping to the CSN-DC Functionalities. For non ACS owned COTS the reference release has been indicated.

COTS	CSN-DC Components	Note
Liferay 6.2	WUP (WCMS) COM	Liferay is used to perform all the COM operations and the CMS for the WUP. Alfresco was dropped as the CMS provided by Liferay covers all the requirements.
ORACLE Identity Manager	UMA	User provisioning
PDS-DGF	IIF PDE	Data exchange with external providers Product delivery
PDS-Thin Layer	PMA	The PDS-Thin Layer is used for Near Real Time production processing chains
ISM	DAM archive	The ISM implements the archiving facility of the DAM
MAT	POR (part)	A Mission Planning tool to manage more easily and with more efficiency the interface with various Satellite providers

SIBILLA	GisViewer, POR and Alerting	SIBILLA is the framework used for developing the WUP applications.
Degree2, version 2.3	CSW/WFS	Based on open source software extended, partially reworked and deeply optimized by ACS to provide CSW/WFS OGC Service
GeoServer, version 2.6.4	WMS	Based on open source provides the WMS service
Jasper 4.6.0	Alerting	Used to generate all the needed reports.

3.3 Components developed/adapted

CSN-DC Components	Existing Components
GISViewer	<p>The WUP components are largely based on OGC WMS/WFS/CSW clients that manage data requests to a spatial-enabled service and make data accessible through http protocol.</p> <p>Over these elements an exhaustive web interface for querying archived data has already been developed by ACS in the frame of an Oil Spill monitoring systems such as Italian National Oil Spill monitoring service PRIMI.</p>
Alert Manager	The Alert manager is a fundamental component used to generate alert reports.
POR	The POR is based on a state-machine environment accessible via the web, which allows various actors to interact for the end-to-end image allocation and purchase. The POR is capable of displaying the satellite scenes over a map together with their metadata.
JOU, FINSYS	Developed ad-hoc for the CSN-DC

3.4 Internal interfaces communication mechanism

Most internal interface use communication mechanisms based on the JMS queues exploiting the Weblogic JMS implementation. This approach is simple, robust and compliant with the Weblogic middleware infrastructure deployment. There is no need to implement more complex communication mechanisms, such as the Oracle ESB.

3.5 Introduction to PDS (Payload Data Segment)

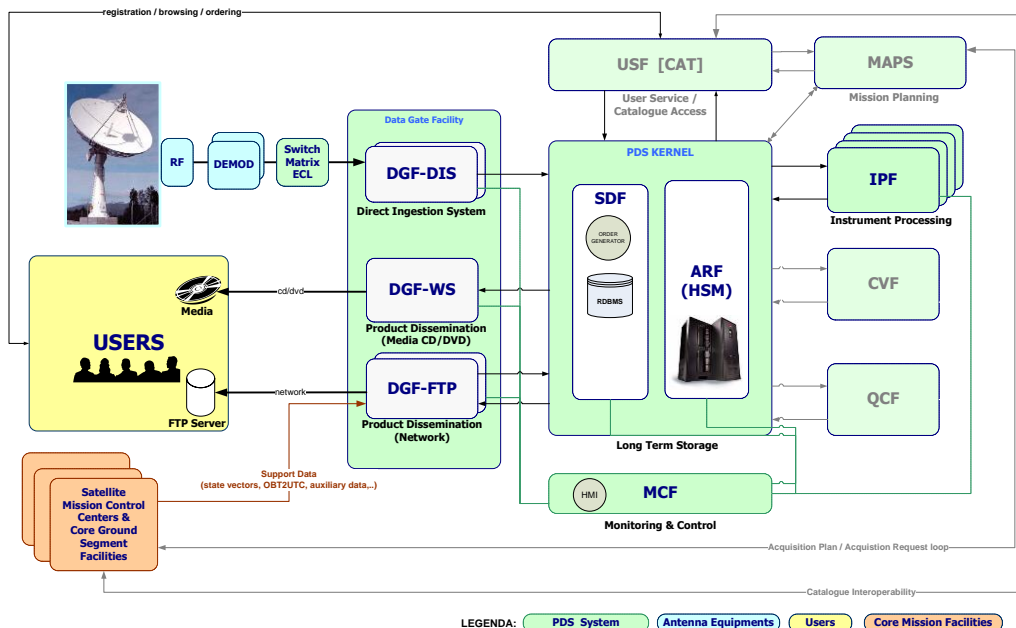
The PDS is by far the largest COTS reused by ACS for the development of the CSN-DC. Due to the highly modular characteristics of this COTS it will be possible to re-use specific parts. In order to have a common understanding of the PDS components functionalities, a short introduction is reported in this section, which allows a better understanding of how the system works and how it is used as a major backbone of the CSN-DC.

The PDS is a System originally designed within the scopes of the “Cryosat Payload Data Segment (PDS) Project”, for the European Space Agency.

It is based on the experience gained by ACS in EO Architecture in the last 20 years: it can host any EO mission and it runs on Linux O.S.

The **PDS** is a Multi Mission Payload Data Segment designed to perform Ingestion, Archiving, Processing and Dissemination of EO Satellite Data.

It is an infrastructure with well defined, easy and not fancy protocols (mostly file based and use of standard RPC (broadcasts, sockets)) to which services and facilities can easily connect to.

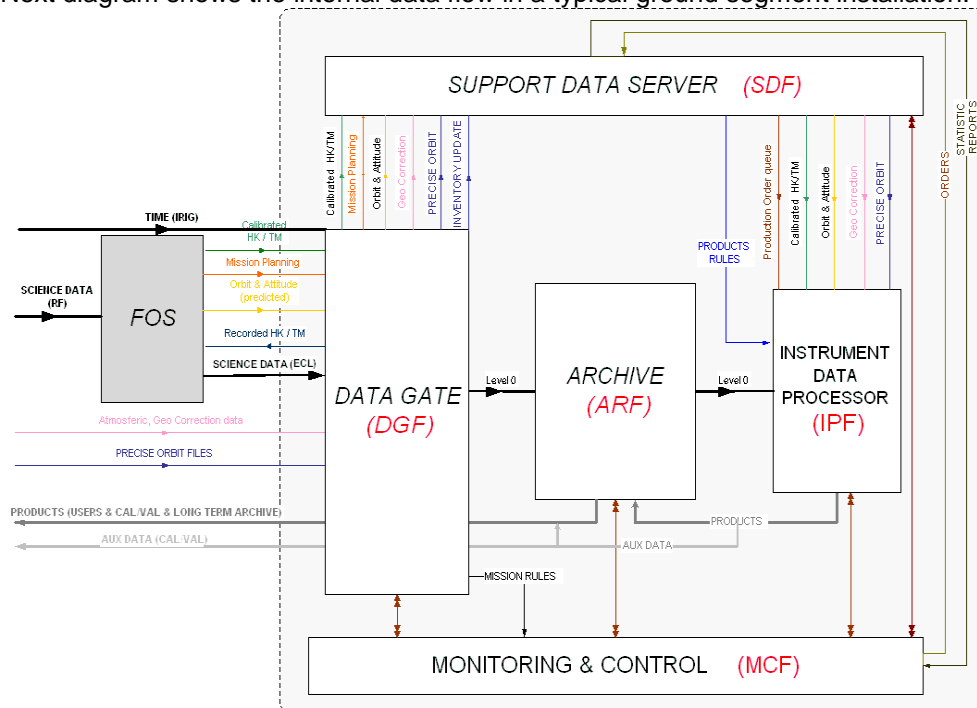


The previous diagram shows the high level view of one of the possible configurations of the PDS in a Satellite Ground Segment.

Main components are (the components which have been included in the tailor made version for EMSA CSNDC are underlined):

- Green blocks: The PDS core components:
 - **DATA GATE FACILITY (DGF)**, managing the internal and external data flows from and to PDS;
 - **ARCHIVE FACILITY (ARF)**, managing online and near-line data storage;
 - **SUPPORT DATA SERVER FACILITY (SDF)**, managing the database handling of the different tables, logging and storing all files ingested and produced by the PDS, including all support files for the production,
 - **INSTRUMENT PROCESSING FACILITY (IPF)**, managing data processing,
 - **MONITORING & CONTROL FACILITY (MCF)**, managing a centralized monitoring and a “very light” control of the overall PDS system.
- Green dotted blocks are optional components:
 - **CVF**: the calibration and validation facility for the satellite data
 - **QCF**: The quality control
 - **USF**: The user service, that provides web access to stored data and products
- **ECL** Switch matrix to connect the Data Gate Facility to the antenna equipment

the MCF has not been delivered because EMSA has its own monitoring and control systems, while the optional components are not relevant for the specific CSNDC business cases.
Next diagram shows the internal data flow in a typical ground segment installation:



3.5.1 Data Driven

Often “Data Driven” working mode for an Earth Observation Facility is wrongly only associated to a system able to perform a “systematic” production.
The implementation of the Data Driven in PDS has a much wider meaning.

It is true that it is possible to configure a systematic production based on policies, defined per File Type (e.g. “when the Storage Archive receives a File Type ‘k’, queue a production order for Processor ‘X’ with priority ‘N’”) but the system is also able to receive and queue a production order from an external interface or to allow the operator to insert a manual order (On Demand Production).

Therefore, the classical “Data Driven” approach has been extended in the PDS to include these possibilities.

Since it does not deal only with “systematic activities”, the PDS has been empowered with “wake up triggers”, aiming to wake the system on the “Event that new Data becomes available”.

It can be seen as a hybrid solution between an Event Driven and a Data Driven System, although for its natural way of activating, carrying on parallel actions, distributing the intelligence rather than centralising it, it is naturally surely not belonging to an Event Driven system.

With these considerations in mind, ACS has designed a system able to react very rapidly on events that, for their nature (i.e. new data available) can be referred as “Data Driven ++, a Distributed Intelligence Architecture” .

In fact:

- It has a shortened latency (i.e. sub-elements wake up immediately as soon as new input data is available)
- It has no Time-programmed actions to be designed
- It is easy to be integrated as all interfaces are simple and based on XML
- It is easy to expand to heterogeneous elements (e.g. other processing chains have to be integrated with minimum effort)
- It has a light controller approach (no overall central “intelligence” but rather distributed logic)
- It has a prioritised processing to manage systematic and on demand production orders in the various modalities foreseen
- It has a distributed architecture against centralised decisions
- Its scheduling policy is based on tasks allocation made by each available node, rather than on predefined assignment of nodes to a predefined task

Data Driven in the Production System: Capability of the processing nodes to wake up in the moment all prerequisites for the execution of a job production are met (e.g. all mandatory input files with specific matching criteria, timely or geographically based, become available, such as QNO + EOP packages necessary to produce the PV image).

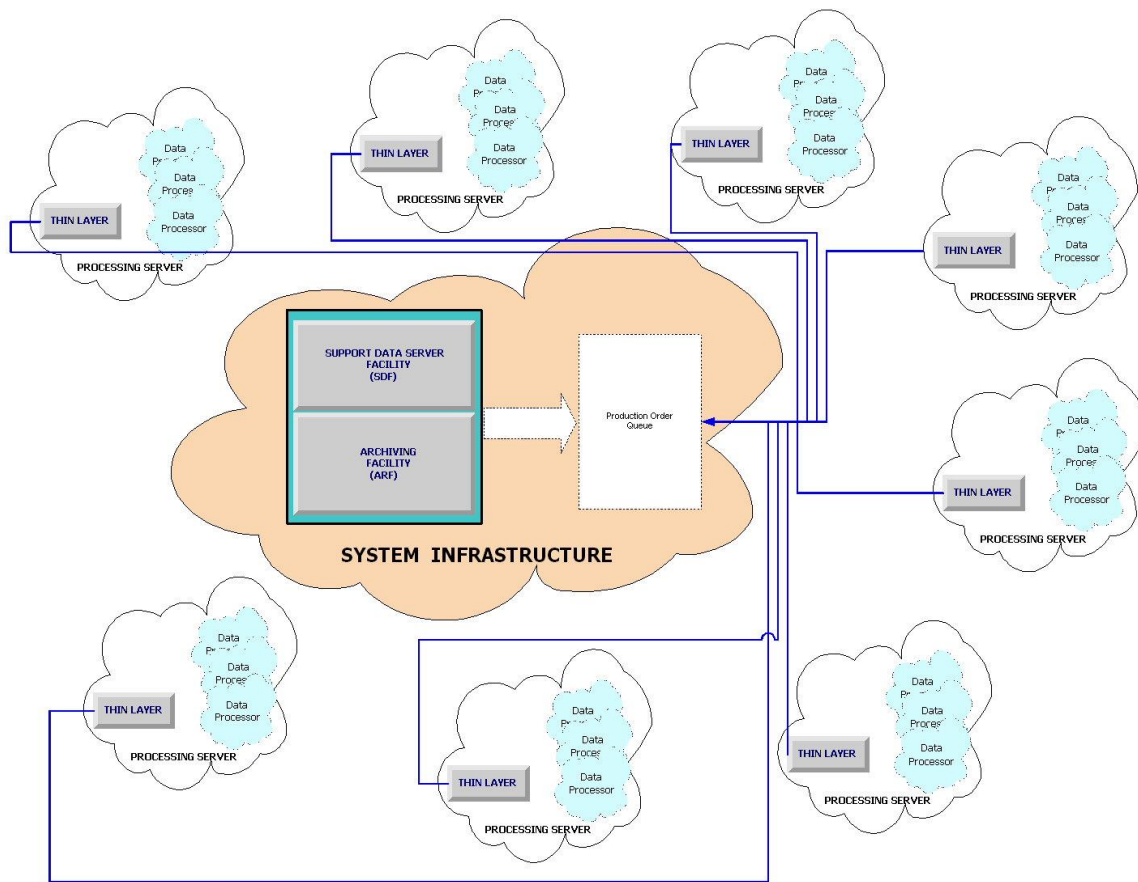


Figure 3-1 Data Driven in the Production System: the Thin Layer

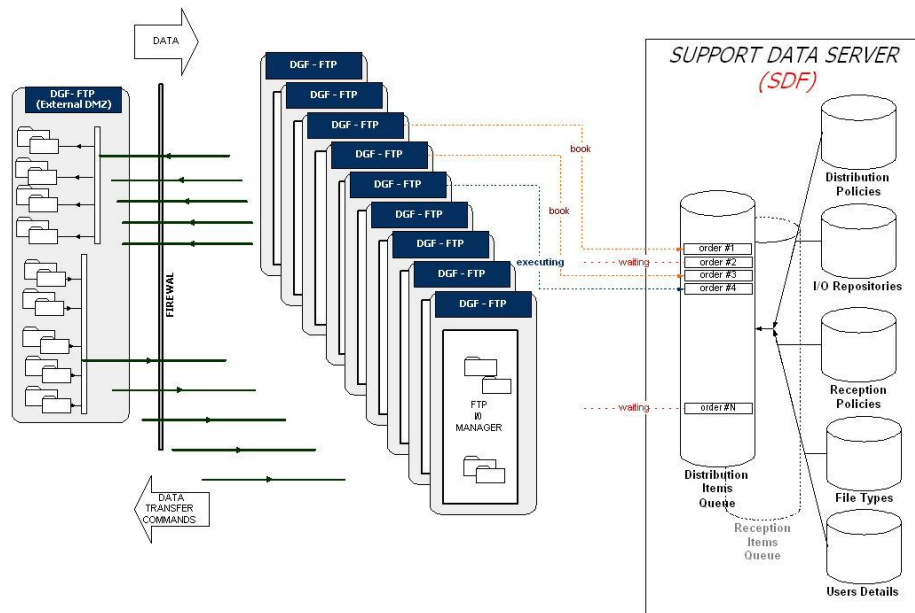
Processing System is not “controlled” by the PDS infrastructure or Monitoring and Control Facility but it is rather the opposite, as the infrastructure “exposes” a production queue and does not know anything of the Processing System deployment (e.g. number of machines, IP addresses, capabilities).

Processing System machine in the LAN, via the Thin Layer, polling the production queue, according to the processor installed books, verifies the existence of prerequisites for the job execution, downloads the input files, performs the processing, updates the status of the production order and finally saves the results in the Storage Archive.

Also the machines composing the Processing System do not know the existence of other machines in the LAN and, simplifying, “compete each other” to book a new production order.

Data Driven in the **Distribution System**:

Network or Media files distribution is regulated by “**Distribution Policies**”.



As the Thin Layer does for production, any Distribution subsystem installed in the LAN (one or many) sleeps and wakes up in the moment a new order for distribution becomes available, checks the queue and takes over the job ("against" other instances of the same dissemination method e.g. media or network) to disseminate the file to the configured User.

3.5.2 Modularity

Thanks to the Data Driven approach, PDS Facilities are almost independent of each other. As already said, PDS Kernel is composed of SDF+ARF, namely the Storage Archive. All other facilities (DGF-FTP, DGF-WS, DGF-DIS, IPF) can be deployed or not without even reconfiguring the PDS Kernel. This high level of modularity allows the PDS to be deployed in various instances according to the specific site functional requirements with SW modules modification (as an example, see PDS installed for CryoSat at LTA Site (CNES) where the PDS Kernel is used with IPF to act as reprocessing centre) .

3.5.3 Scalability

HW Configuration on which the PDS can run is extremely scalable.

IPF Science Processor can be hosted in a number of machines/servers/workstations completely transparent to the PDS thanks to the Thin Layer architecture.

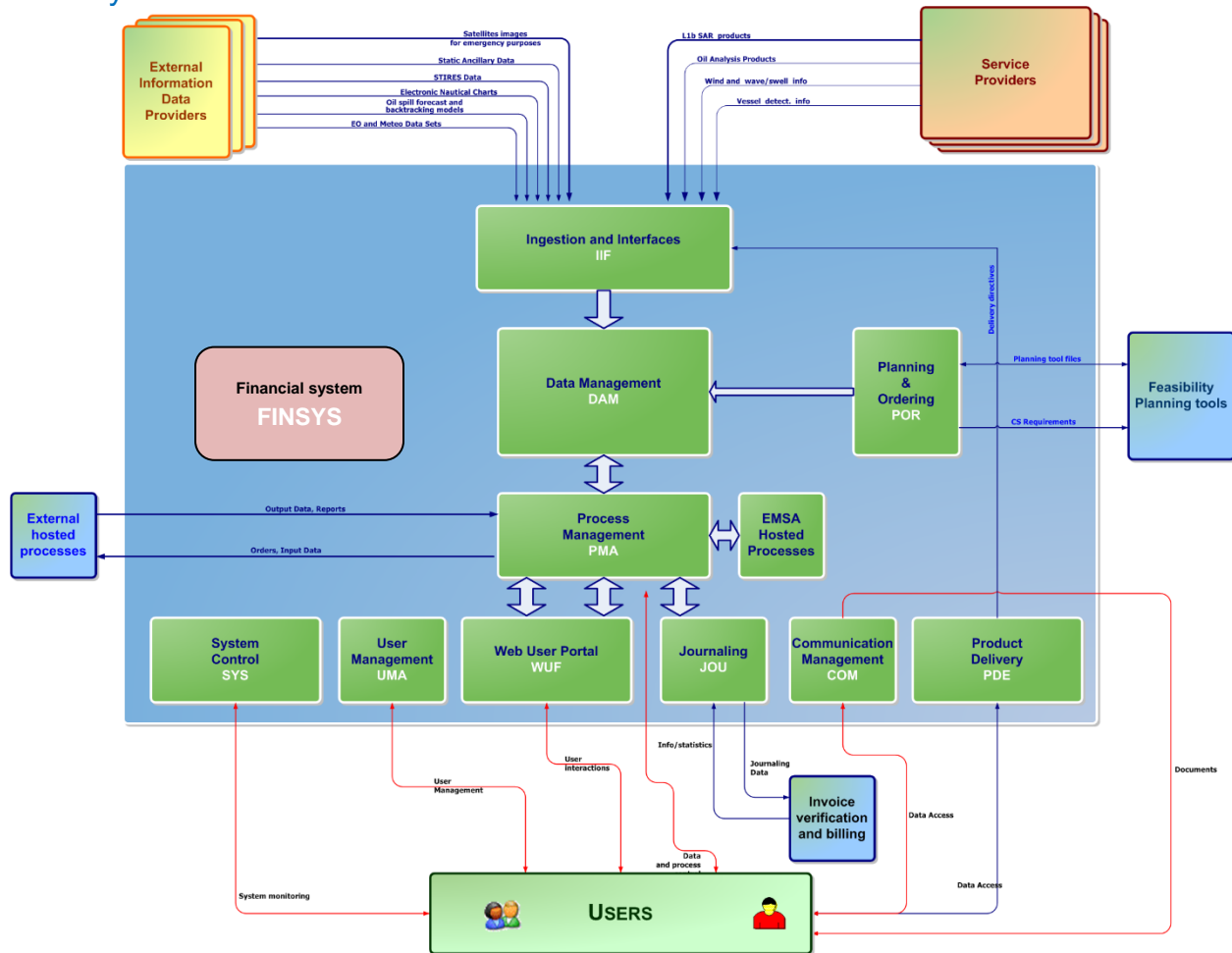
No reconfiguration is anyway needed in case the number or processing machines is incremented during the lifetime of the system.

N.B. growing factor in terms of throughput is almost linear adding a new server (up to the maximum bandwidth of the PDS Core)

4 Logical View

This chapter describes the elements and services that compose the system, showing its structure and the way they interact.

4.1 System Overview



4.2 DAM – Data Management

4.2.1 DAM Overview

DAM is in charge of providing the basic platform for data exchange and storage. The DAM architecture is illustrated in following figure

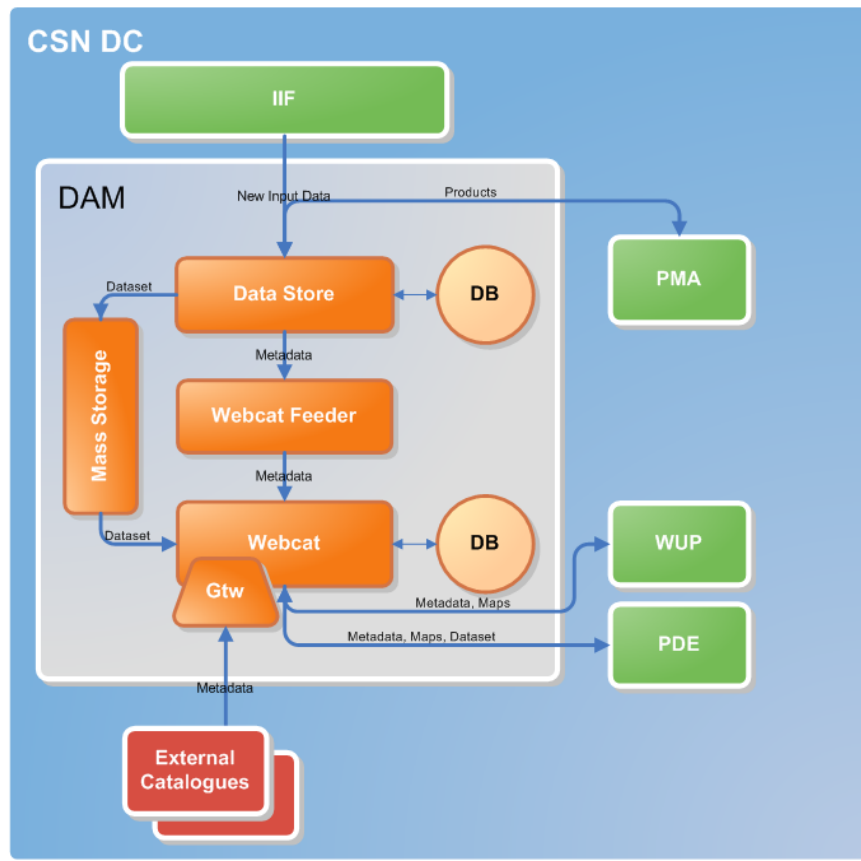


Figure 4-1 - DAM high level architecture

Here is a list of typical services DAM offers:

- Provide basic services to IIF for the secure exchange of data and metadata among CSN DC components, in particular:
- Manage the in/out data flows versus the PMA, i.e. making data available to processing functions and storing the output data of the processing functions
- Storage service for external incoming data, intermediate and output products
- Provide long-term and mass-storage services
- Metadata extraction services
- Dissemination services to WUP (to GISViewer, POR and COM interfaces integrated in WUP) and PDE:
 - Catalogues services (e.g. OGC CSW)
 - Raster maps offering services (e.g. OGC WMS)
 - Feature browsing and offering services (e.g. OGC WFS)
 - Coverage offering services (e.g. OGC WCS)

- Support concurrent search (and result set recombination) on different internal (to the Agency) and external OGC CSW and WFS compliant services and predefined external custom catalogues

In its operations, DAM handles different and heterogeneous datasets originated both internally in the Agency and by external service or data providers.

4.2.2 DAM Decomposition

As illustrated in the next figure, the DAM is composed of the following modules:

- Data Store
- Mass Storage
- Webcat (with its gateway to connect remote services, 'Gtw' in the figure)
- Webcat Feeder

The WebCAT also includes the following 2 COTS components:

- Deegree
- Geoserver

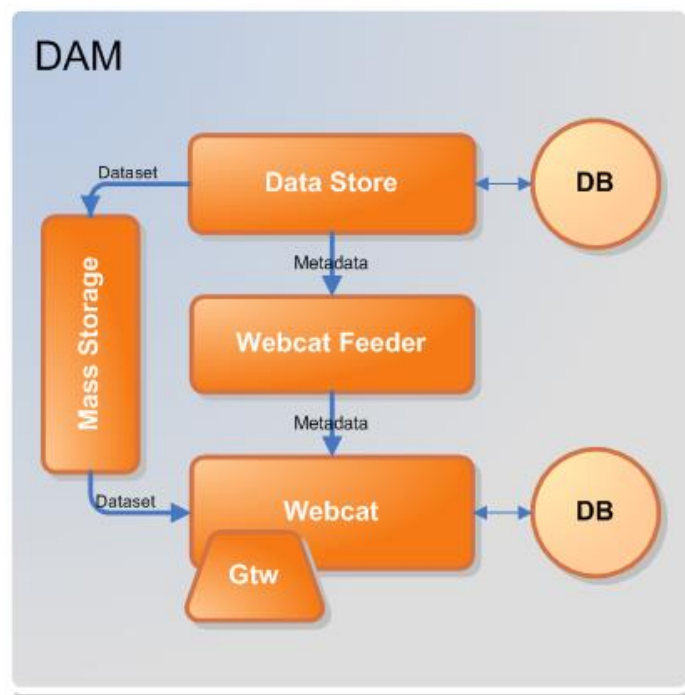


Figure 4-2 - DAM decomposition

4.2.2.1 Data Store

Every ingestion in the DAM goes through the IIF component. Data files coming from the IIF are “inventoried” in a Data Store module. The Data Store module is the main central repository of the metadata extracted by data files during inventory process. It is based on a relational, spatial-enabled

database. Data Store module also acts as data gateway: metadata are never directly accessed through the database. The “internal” data exchange is made through the Task Manager component of the PMA so ensuring the possibility to manage each data exchange in a workflow.

Data Store is based on ACS’s Support Data Facility (SDF) solution for metadata storage in “mission critical” Payload Data Systems (PDS) for EO missions such as the ESA Earth Explorer CRYOSAT, ADM-AEOLUS and GOCE.

Moreover SDF is used in several different Ground Stations all around the globe, as installed in the ACS “Compact Station” multi satellite solution, in a 24x7 operational context.

4.2.2.2 Mass Storage

Mass Storage module provides services for storage and long term archiving of bulk data files.

The Mass Storage module is implemented by ACS’s Intelligent Storage Management (ISM) solution. ISM is the storage and long term management system of ACS used 24x7 in “mission critical” EO Ground Segments such as ESA GOCE and ASI COSMO SkyMed but will be soon for CNES VENUS , ESA CRYOSAT and ESA ADM-AEOLUS each of them with tens of images produced daily.

The current version of ISM installed at EMSA is ISM Lite

ISM Lite can be further decomposed in following sub-modules:

- **ISM WebServer:** It is a standard web server application (Apache) which controls ISM web services. It exposes a SOAP interface that is used by the clients (such as the PDS – Inventory) to coordinate an I/O operation. The physical file transfer operations is instead performed using standard files transfer protocols (FTP)
- **ISM DataStore:** This is the physical storage where ISM Lite stores all files. It can be implemented using a DAS/SAN/NAS disk-based storage
- **ISMClient:** it is a software that uploads/downloads/erases files from the ISM. It links a dedicated C++ library called libISMClient. The library implements a SOAP-based communication protocol with the WebServer. It also handles the file transfer activities.

Since it was tailored over “dual scopes” EO missions, such as ASI Cosmo SkyMed, **it runs also over a certified environment (Common Criteria EAL4+).**

The ISM WebServer is used whenever it is necessary to store and retrieve mass data, such as the CSNDC data packages. It basically takes care of the physical transfer and retrieve of the package files on the ISM store. Typical examples of usage of this is:

- when a data package is ingested by CSNDC
- when a EOP package is retrieved by the PDS EO Processor and when the results of the processing (e.g. the TIFF pyramidal file) are stored again on the ISM
- when the feeder reads the data to perform an ingestion into the OGC services

The ISMClient is normally not used by the CSNDC application. It is a simple utility which can be exploited mainly for debugging reasons (e.g. to troubleshoot issues with the ISM).

4.2.2.3 Webcat

Metadata ingested in the Data Store are also “published” in the Webcat module. Webcat module is in charge of providing OGC Service interfaces to data stored in the DAM.

It exposes OGC CSW, WFS, WMS, WCS. It is based on Java **Open Source** software packages (i.e. deegree2, see § 3.4.4) and implements Collection and Service discovery service, Earth Observation Product Metadata and Catalogue Search services (implementing the Extension Package for ebRIM application profile) and the HMA Ordering services (**req. 5.1.0.1-10**). Web services offered by Webcat module will be used by the GisViewer component to publish data to the user in a rich client environment (see § 3.7). In the same way, Webcat services are used by the PDE components to retrieve maps, data and information to be included in reports (Alert Report) and output products (Standing Orders) for the final user.

Webcat module also has a plugin gateway to split an incoming GetRecords (or GetFeature) query in a number of separate GetRecords (GetFeature) queries to external pre-configured CSW/WFS URLs and merge back results in a unique result set (**req. 5.1.0.11-13**). The gateway can also manage integration of queries on external custom catalogues by developing specific adapters.

Webcat module implements also Transactional CSW services (Delete, Update, Insert).

The Webcat module is realized on the basis of deegree2 Java Open Source software package (www.deegree.org) and Geoserver (<http://geoserver.org/>).

Deegree2 supports the following OGC standards:

- WMS 1.1.1 / WMS 1.3
- WFS 1.1
- WCS 1.0
- CSW 2.0.2
- WPS 0.4.0
- WTS/WPVS (pre-standards)

As another, even more important feature of deegree CSW is that it does not contain a data access module nor a specific datastore model of its own. It uses a deegree WFS as datasource and XSLT processing to transform requests as well as responses into the desired format.

As deegree WFS is extremely flexible in supporting different database schemas, it is possible to adjust deegree CSW to almost any existing metadata structure. This is the basic mechanism that permitted ACS to customize “standard” deegree CSW to a CSW-ebRIM for EO Products (see § 3.2.4 below).

While Deegree alone supports all needed standard, it was chosen to use Geoserver for the raster-oriented type of services, typically:

- WMS 1.1.1 / 1.3
- WCS 1.0

The choice of Geoserver was done after a technical trade off, which demonstrated that it is more suitable for the raster oriented processing.

The feed of the various OGC services is performed using different strategies, depending on the data type, the service type and also considering throughput and performance needs. As a default strategy an OGC standard operation has been used to feed the data into the services. For the most throughput intensive cases, related to the ingestion of vessel traffic data received from other systems (e.g. AIS data from IMDatE) and detected vessel information, for performance reasons, it was decided to skip the overhead of standard OGC transactional requests and store the data directly into the database appropriate tables. The WFS-T operations for these data is still available and fully functional, but it has lower performances mainly due to the need of managing data in a less compact format.

The following table reports a mapping between data type, package type, OGS service handling the data and data ingestion approach used.

Data type	Package	OGC Service to expose the data	Ingestion approach	Details
Oil Spill	OSN	WFS	WFS-T	Ingestion is performed in 2 steps: <ul style="list-style-type: none"> transformation of the original oil spill XML into another XML, suitable for ingestion into Degree (using xslt transformation) Direct request to the WFS-T providing the transformed XML in input
EO Metadata	EOP	CS-W	CSW-T	Ingestion is performed in 2 steps: <ul style="list-style-type: none"> transformation of the original oil spill XML into another XML, suitable for ingestion into Degree (using xslt transformation) Direct request to the WFS-T providing the transformed XML in input
Detected vessels	DER	WFS	Direct database insert	Data are extracted from the original XML and are inserted into the database, more specifically into the DSIMAGEIDENTIFIER and DETECTEDSHIPS tables.
Vessel traffic data (e.g. AIS from IMDatE)	Retrieved from IMDatE	WFS	Direct database insert	Data are extracted from the original XML and are inserted into the database, more specifically into the AISVESSEL and AISTRACK tables.
Raster images	PV (generated internally by the CSNDC processor)	WMS / WCS	Geoserver APIs	The feeder uses the Geoserver REST APIs (http://docs.geoserver.org/stable/en/user/rest/api/) to create the layers. The following 3 steps are performed: <ul style="list-style-type: none"> creating Workspace (if not already existing) creating a store creating the layer

Table 4-1 – Detailed approach used for feeding the data for the various OGC services exposed by CSNDC

4.2.2.4 WebCat Feeder

Metadata in the Data Store are “published” in the WebCat thanks to a dedicated module called WebCat Feeder. WebCat Feeder is continuously checking for any change event in the Data Store repository. As soon as an Insertion, Update or Delete event is detected, the Feeder extract metadata and push them in the WebCat. A link to the original dataset stored in the Mass Storage is maintained in the metadata for later access and retrieval. Publishing of metadata is automated and configurable through policies and rules allowing for a conditional metadata publishing.

Please note that Webcat Feeder module, along with the Webcat module, has been already successfully used in a number of projects such as VENUS, Centro Nazionale Multimissione CNM, PRIMI.

4.2.3 Geoserver logical architecture

A specific solution was chosen for the implementation of the clustering for Geoserver, which is based on the following characteristics:

- based on Tomcat as application container
- running on specific independent machines (named [x]GEO[yy], where x stands for the environment, e.g. t/q/p for test/pre-production/production and yy is typically 01 and 02), which are independent on the servers running WebLogic

The solution chosen for the Geoserver cluster deployment is based on a scenario developed by Boundless (<http://boundlessgeo.com/>) with the support of Geosolutions (<http://www.geo-solutions.it/>).

The general features of this solutions are the following:

- Geoserver is running inside Tomcat
- each cluster node is implemented on a different machine (each running Tomcat + Geoserver)
- a load balancer machine is used for managing the cluster
- a *jdbc-config* plugin is used in order to store configurations (e.g. <http://ares.boundlessgeo.com/geoserver/2.6.x/community-latest/>)

The last bullet is one of the key of the success of the solution. In fact in a standard installation Geoserver reads its configuration from the so-called DATA_DIR. In a clustering configuration, this data directory (which normally does not contain the bulky data, but only pointers to the data) is accessed via a network sharing (e.g. NFS), but it could be a bottleneck is multiple nodes are reading/writing from this folder in parallel. With this plugin the data directory will have the option to be database-backed. This means that a central configuration store can be queried more optimally than a file-based counterpart and doesn't all need to be read into memory.

The deployment is depicted in the following figure.

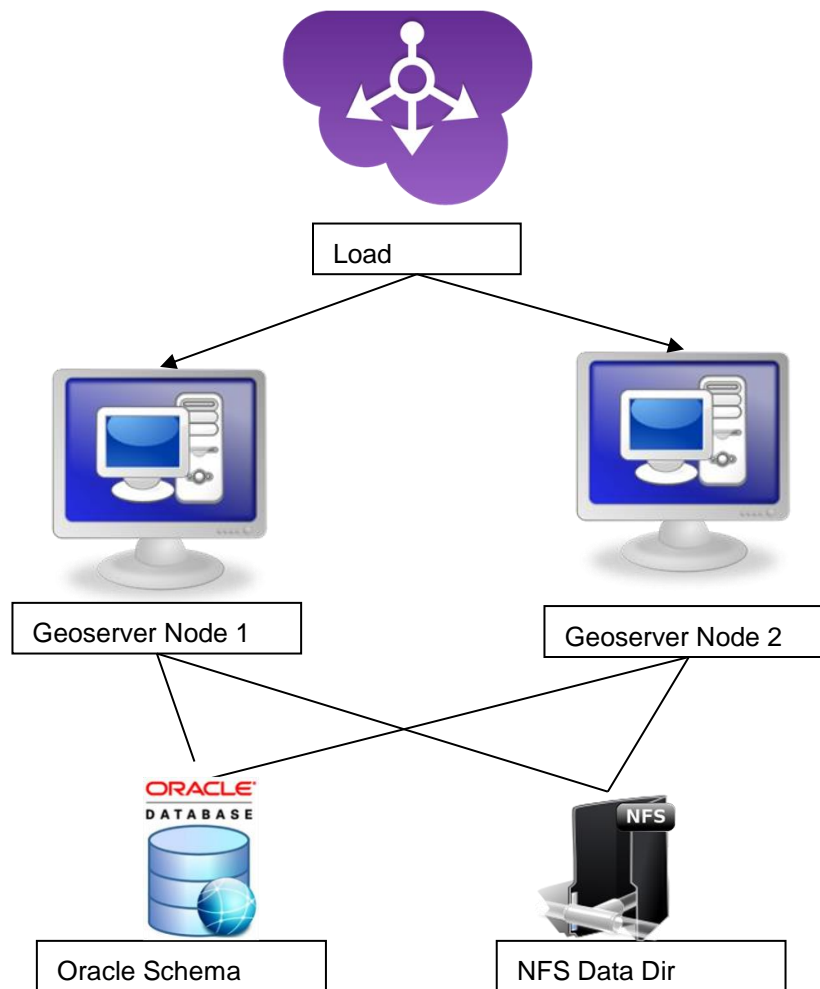


Figure 4-3 – Geoserver deployment

In Geoserver Nodes, the following is installed:

- Tomcat Version 7.0.55
- JAVA Version 1.7.0
- Geoserver Version 2.6

Oracle Schema: A new schema on the Oracle RAC, which will be used by the JDBC config plug-in. The New user will be named GEOUSR and will have some schemas adapted for storing the configurations.

4.2.4 Implementation of the Catalogue Services

Catalogue Services are defined in OGC specifications as services for the discovery and retrieval of spatial data and services metadata. More specifically, the OGC Catalogue Services 2.0.2 specification (OGC 07-006r1) establishes a general framework for implementing catalogue services based on the HTTP protocol binding (CSW).

Catalogued items (EO scene, oil spill, detected vessel...) are represented by a specific GML metadata structure.

OGC Catalogue services do actually deal with the specific GML metadata structure in two main circumstances:

- Filter encoding/decoding while discovering resources (e.g. in a GetRecords request). In facts, managing the specific structure of GML metadata schema is necessary to express a set of filtering criteria.
- Transaction operations (insert, delete, update)

A specific profile of the OGC CSW joins the CSW interfaces to the OASIS ebXML registry information model (ebRIM 3.0) with the aim *“to provide a general and flexible web-based registry service to locate, access and make use of resources in an open and distributed system”*. This kind of Service is generally referred to as “CSW-ebRIM” Registry Service.

Such a Service, in facts, provides facilities for retrieving and managing many kinds of abstract resource descriptions. An extension mechanism permits registry content to be tailored for specialized application domains. Specifically, it is possible to define how EO products are organized in a Catalogue by describing a mapping of EO metadata to an ebRIM structure implementing the CSW-ebRIM Registry Service. Such a tailoring (that, strictly speaking, does not represent itself a “specification” but rather a “best practice”) represents the so called “EO Product Extension Package” (defined in document [HMA-CAT]).

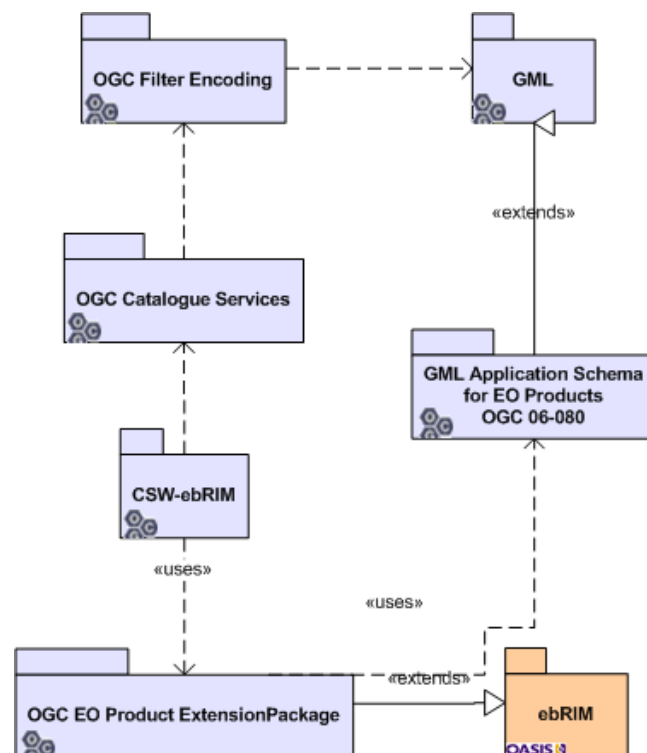


Figure 4-4 - EO Catalogue specifications relationships

4.2.5 Specific configuration of Deegree to implement the ebRIM scheme and the EOP profile

This section provides some more technical insights to explain how ACS has configured the Deegree to implement the EOP profile using the ebRIM data model (as explained in the Figure 4-4 above). The configuration is rather straightforward, as Deegree comes out of the box without a specific data model but allows to implement a given data model by performing the following tasks:

- Creating a specific database structure
- Configuring appropriately the Deegree COTS configuration files (typically transformation schemes) in order to have Deegree correctly interpret the data from the database schema and respond to service requests that will adhere to the appropriate profile ([HMA-CAT] in this case)

For more details about the database schema please refer to § 8.1.1. The database model is mostly driven by the very data of the service responses as specific in [HMA-CAT]. This agnostic structure, based on the ebRIM data model allows to easily add new attribute in the future which were not initially planned. On the other hand, the database schema has been partly tailored to the specific needs of the EOP data searches, trying to optimise the search for certain specific parameters that are mostly used. This is evident by looking at some EOP major attributes (e.g. beginPosition, endPosition, productType) that are most frequently used when querying the data. It is important to highlight that the CSW services responses are fully compliant with [HMA-CAT] specification and that the internal structure of the database is completely up to the implementation details of the solution.

On top of the standard Deegree configurations described above, another task which has been necessary in CSN-DC is to implement a porting on Oracle, which was implemented by integrating the low level appropriate libraries.

4.2.6 Implementation of the ingestion of raster data

Raster data are a fundamental component of the CSNDC which exposes SAR (and also Optical in the latest releases) images made accessible to the GIS Viewer via standard OGC interfaces. The management of such layer is quite simple, as they are ingested into Geoserver using the REST APIs made available by the COTS. After ingestion, the raster data become available in Geoserver as raster layers and are simply accessed via a GetMap request from the Web Client (the WUP GIS Viewer in this case). That can be also requested using standard OGC calls, e.g. GetMap thus allowing external system to access these data.

4.3 IIF – Ingestion and Interfaces

4.3.1 IIF Overview

The IIF component has in charge the two-way exchange of data between the CSN DC and any external data file providers.

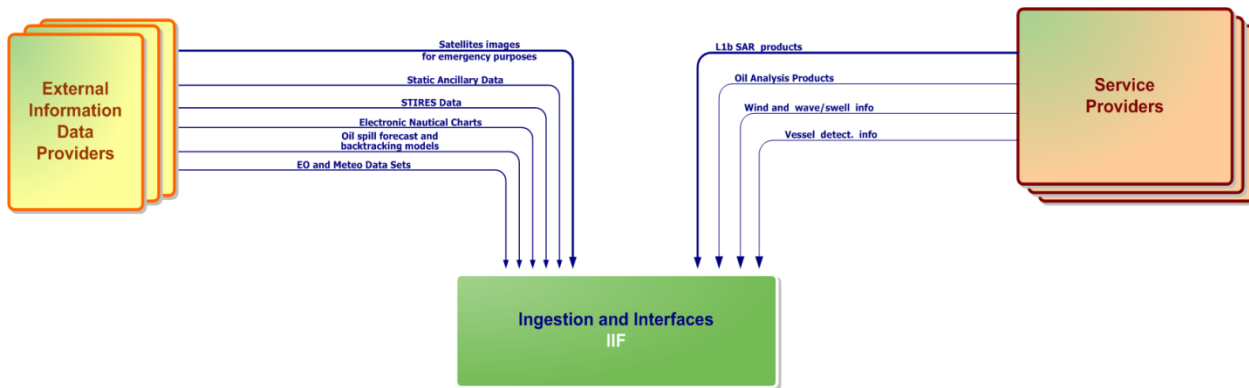


Figure 4-5 – IIF as the CSN-DC file-based data gate

It acts as the data gate for all files exchanged with the external world of the entire CSN DC.

IIF implements that both Import and Export of files over the network.

It works on both **systematic policies** and **on demand request**, done by other components of CSN DC (e.g. PMA) (**req. 6.0.0.5**).

Both Import and Export functions, independently whether they are triggered on systematic or on demand requires an URL (repository) to be reached.

Moreover, for the systematic policies, each URL requires also some other parameters such as “polling frequency”, preferences on the data transfer mode “push or pull”.

The keystone of the management of the file-based interfaces (internal and external) by IIF is the concept of **File Type**.

Each time a file is ingested/produced in the system, it needs to be archived into the Storage Archive (i.e. DAM). The first step, before archiving, is the **analysis** of the file to **assign it to a specific File Type**. This association shall after induce a number of actions that the system shall perform on the file itself, after its archiving. The way to **recognize to which File Type a file belongs to is based on a “regular expression”**.

The “**Name Regular Expression**” (s) defined in the system are tested one by one on the file name up to the moment (the first match) the file is recognized.

In case the file cannot be associated to any configured File Type, the System behaves depending on the different application/component (e.g. in case of FTP Import the file is simply ignored).

Pre Inventory and **Inventory** tasks specification is a very useful feature when a new File Type is defined, having a completely new format that, in order to be inventoried (i.e. archived), need to have a dedicated “metadata extractor” that in the proposed architecture can be concentrated in simple dedicated tasks (the Pre Inventory and the Inventory). The system has already a number of PreInventory/Inventory programs developed managing a mandatory XML header defined for the ESA Earth Explorer File Format Guidelines, the ENVISAT GS file formats and other mission/project data formats.

4.3.1.1 Integrity, Format and Quality control checks

At **PreInventory** level, the ingested files are “processed” in the following way:

- Execute the formal **consistency/integrity check** function to control:
 - The correct size of file;
 - The coherence with specific product structure.
- If the formal check fails, the file is rejected: the error is logged and the file is stored in a directory dedicated to these files. The processing is interrupted and no storage is executed in Inventory or archive.
- If the formal check has a good result, it means that the pre-inventory is able to extract metadata for the file inventory and that the file can be archived. Hence, the function extracts from the product all the parameters necessary to fill up and generate the XML file with metadata.
- Execute **simple quality control on headers content** and set the quality control flag. For example, in case of an EO Product the controls executed are listed in the following:
 - File name equal to header related field
 - Validity start/stop times contained in filename equal to header related fields
 - Validity stop > validity start
 - Processing stage flag contained in filename equal to header related field
- Send the XML metadata file created to the Inventory and the input file to DAM

All files ingested in IIF by external entities and/or by operator manual uploading (after an interactive modification) will have specific internal DB file types; generally the file type is represented by the first ten characters of the file name but no strictly rule exists.

4.3.1.2 Auxiliary files Timeliness Verification

The IFF is capable also to **check and verify the timeliness and the expected periodicity of the files to be imported** and raises alarms when the periodicity is not met.

This feature is particularly important for PMA when a workflow has to be activated “data driven” i.e. upon arrival of a given file.

4.3.2 IIF Decomposition

It is worthwhile to immediately state that this component is directly derived by the Data Gate Facility as developed for the ESA EE PDS Systems.

Please note that, in order to minimise the figures clutter, the internal interfaces with the SYS (for monitoring and control) and with the UMA (for identity management) being of general type have not been reported in all the graphs describing the decomposition of individual components.

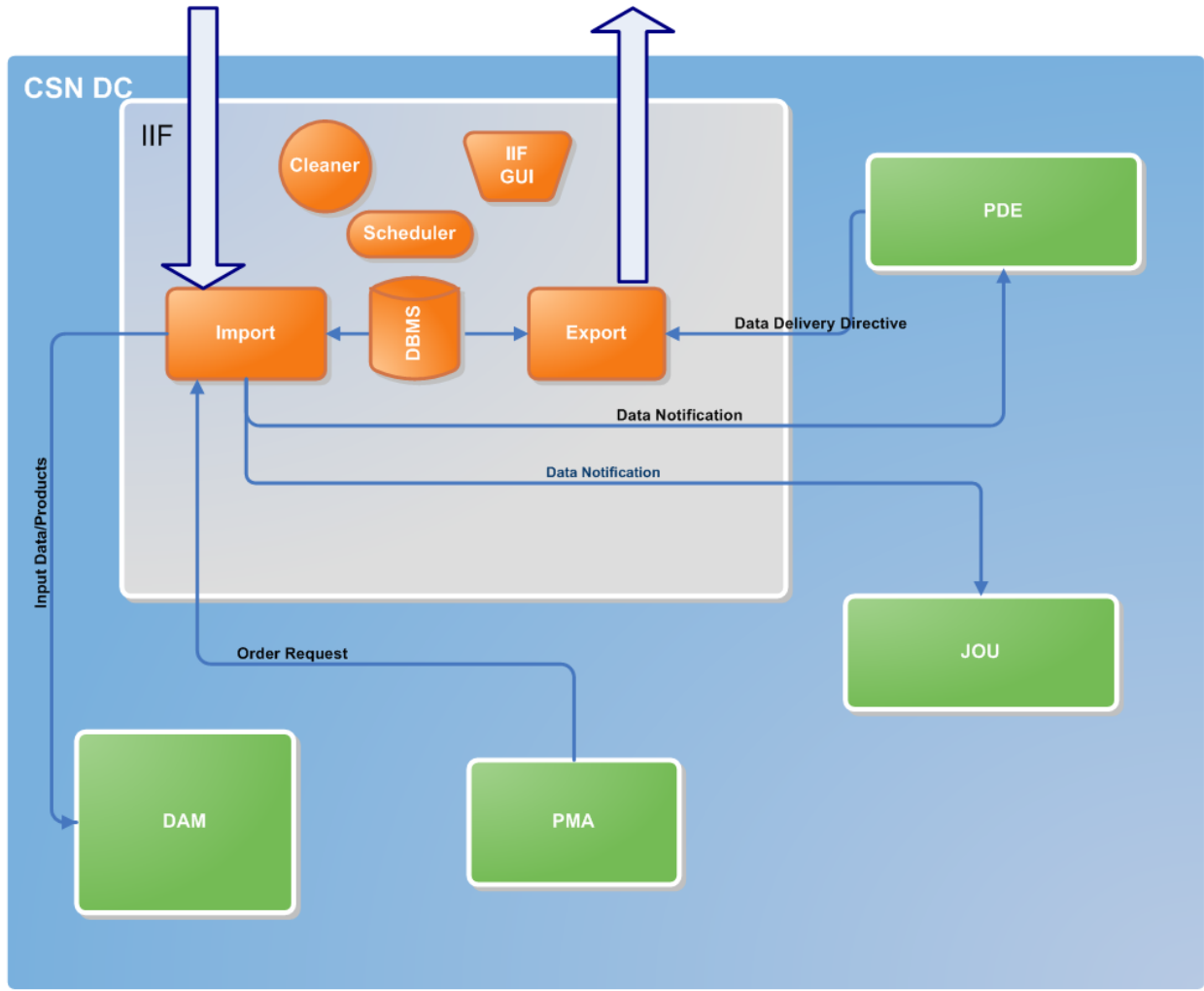


Figure 4-6 – IIF Decomposition

The description of the major components is reported hereafter.

The Import daemon

The Import service is implemented by a Linux daemon which controls all the import operations. It uses some helper applications to perform operation on the external repositories:

- RemoteFileSystemMonitor: is a script/application used to retrieve the list of the files from the remote repository.
- FilterFilesList: is an application used to filter the files returned by FilterFilesList. It's role is to decide which files are to be downloaded.
- RemoteDownload: is the application able to actually download file from remote repositories.

All the helper applications accept a number of configuration parameters to cope with different kind of repositories. Nevertheless it is also possible to write custom applications dedicated to particular cases.

The Import daemon performs **systematic activities** according to configuration policies saved in the database. Such policies specify parameter like:

- The URL to poll,
- The File Type to download,
- The polling frequency,
- Name and parameters for helper applications.
- The PreInventory program to extract the information from the files to fill the metadata needed by the archiving subsystem.
- The Inventory program to save metadata into the archive.

At the same stage the Import daemon manages **on demand import orders** as queued into its collocated DBMS by other components of the system (e.g. PMA workflow).

Systematic activities and on demand order shall be also referred hereafter as “import actions”.

For the implementation of the interface with the Service Providers (SP), the IIF shall also perform the following:

- Expose a web service for receiving the message about the start of data package delivery from the SPs
- Notify the JOU of the reception of the data packages from the SPs

This is described in better detail in § 5.2.

Import manages FTP/SFTP protocols and exercises import actions one by one or in parallel (multithread application). The number of parallel jobs exercised is configurable according to hardware resources and by the network deployment.

Every import action can also specify if the **Import** daemon shall work in “pull” or “push” logic, i.e. if the file has to be downloaded from the site or if the data source autonomously copies the file into a dedicated input basket.

The Import function provides a **fully hot redundant and load balancing** mechanism: it is possible, in fact, by design, to have more instances of the Import function on different servers. Any import action can be managed by one of the N instances of the servers where the demon is running.

The hot redundancy provides a mechanism where each specific import action is “booked” at run time by one of the N servers in the LAN having an Import daemon running.

In case a FTP server fails (i.e. hardware problem), no reconfiguration of the other nodes running the same functions is requested as they will automatically take over the responsibility of all lasting import actions.

The Export daemon

The Export service daemon is in charge of disseminating files towards external I/O repositories i.e. URLs.

It uses a helper application, the **RemoteUploads**, which is in charge to actually uploading files into users' repositories. RemoteUploads accepts a number of configuration parameters to cope with different kind of repositories. Nevertheless it is also possible to write custom applications dedicated to particular cases.

The Export service uploads the files according to the content of a database table: the Distribution Queue. This queue is automatically filled by the PDE for **systematic dissemination policies**, e.g. when a new file is ingested into the system, when a file passes the quality check, and similar events but it can also be filled by **on demand dissemination orders**, again instructed by the PDE workflows (see also § 4.7 and 5.6).

URL for dissemination can be configured to work either with “push” or “pull” logic, i.e. if the file has to be uploaded directly into a remote user repository or if it has to be copied into a local repository (in a DMZ for security purposes) from which the user can download it later.

The FTP dissemination of a file can be directed by a Systematic Policy but also by an “On Demand” external request, received by an external facility.

Export manages FTP/SFTP protocols and, in case of network failure, is capable of performing autonomously more attempts (configurable issue)

The Export is capable of instantiating several FTP operations in parallel, each of them controlled by a dedicated thread. This mechanism allows reducing the time needed to activate the transfer and provides also better group performances: e.g. if a connection remains blocked due to problems on the remote FTP server, the other operations are not affected.

Moreover, the Export service provides a **fully hot redundant** mechanism: it is possible to have more instances of the Export on different servers. Any dissemination action can be managed by any of the N server where the export demon is actually running.

The Hot redundancy provides a mechanism where each specific job to carry on by the Export is booked at run time by one of the servers.

In case a FTP server fails (i.e. hardware problem), no reconfiguration of the other nodes running the same functions is requested as they will automatically take over the responsibility of all lasting export actions.

As one can deduce, the architecture of the Import and Export services are very similar. For this purposes, next picture, providing the redundancy of the Export architecture can be assumed to present also the Import one.

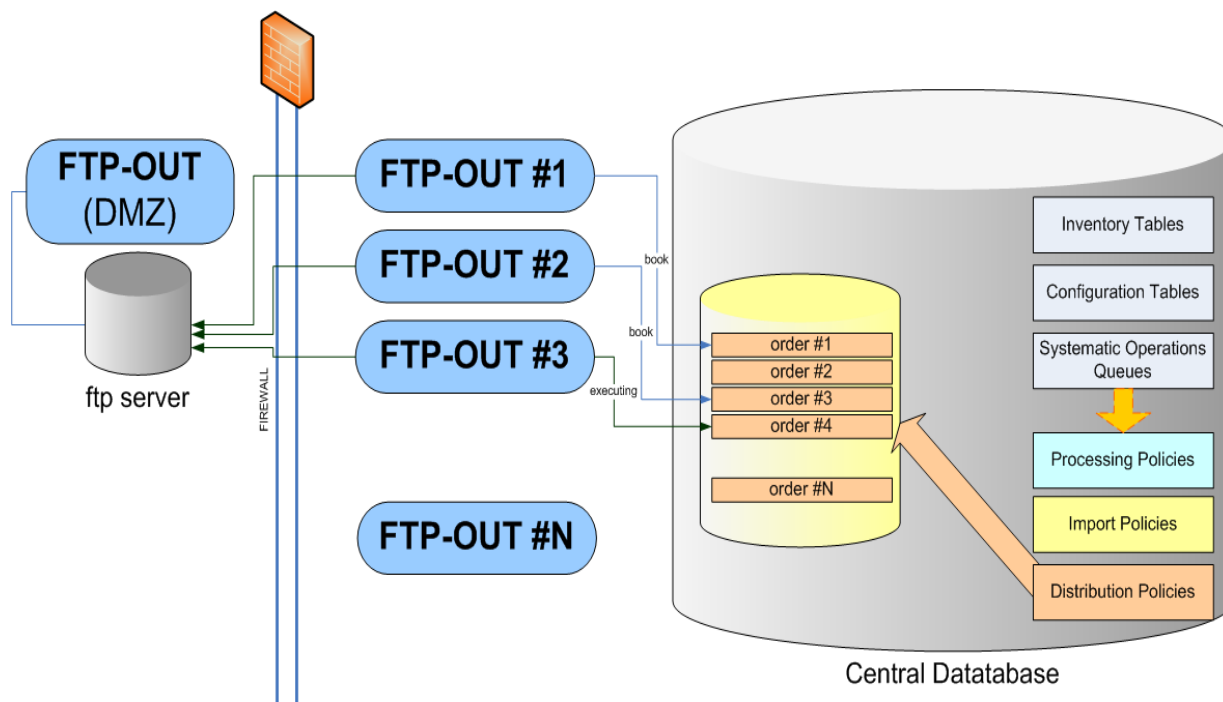


Figure 4-7 – FTP with redundant servers (export/dissemination case)

The Cleaner Agent

Local (DMZ) IO repositories can be configured to be automatically cleaned according to clean-up policies. This activity is performed by a dedicated service daemon, the **Cleaner**.

The Cleaner is configured by the IO repository table where, for each repository, is possible to specify a cleanup policies. Possible options are:

- No automatic cleanup.
- Files older than a configured time are deleted
- Files are deleted when the repository occupation rises above a configured value (in MB)

Each policy can also be activated as recursive, i.e. it scans subdirectories of the repository.

4.4 UMA

4.4.1 UMA Overview

The User Management (UMA) component will implement all functions to manage CSN-DC users, including authentication and authorization profiles. It is a rather complex component as it is involved and responsible of a number of highly heterogeneous functionalities. UMA shall allow different types of users, both belonging to the EMSA and its business partners, to interact with the system: each user is entitled to perform some actions according to his/her own profile (e.g. system administrator, routine operator etc); profiles are organized in different groups with further detail level represented by the roles. Some of these interactions impact critical data, either involving products or user details, and might require an ad-hoc enforcement to be sure that the right person can see the right thing at the right time; some resources might be restricted to a set of users according to their role: when roles change authorizations are supposed to change and to be propagated across all the systems.

From this simple high-level descriptions can be easily identified the core activities in charge of the UMA subsystem:

- Access Management
- Identity Management

Access and Identity management are the complementary activities involved in each access request to a given resource: both of them are to be managed and controlled within the UMA component.

As the number of the involved actors increases (i.e. the resources to be accessed and the users or the applications requiring access either internal or external to the EMSA network) a lot of issues to be addressed arise: with the increasing of the users proper handling of provisioning and de-provisioning becomes more and more important as well as the possibility to customize the workflow of the creation of new users; with the growth of the user-base come the needs of a clean and robust role and privileges management to be sure who's entitled to access a given resource and to track all the activities to increase security and to ease internal auditing.

User Management in CSNDC is represented by the following high level picture.

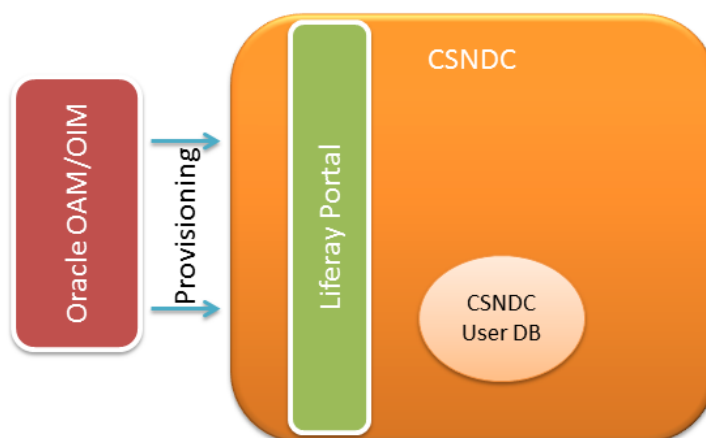


Figure 4-8 Overview of the User Management in CSNDC

Most of the complexity of the user management is handled by an EMSA Leverage system which is based on the Oracle Identity Management (IdM) suite. Since this is entirely managed by EMSA, description of such component is out of scope of this design document. The main concept are highlighted here.

- User provisioning is performed using the Oracle IDM which takes care of all user management aspects related to:
 - Creation, deletion and editing of users
 - Management of user details and user roles
 - Managing the workflow that interfaces with Liferay and the CSNDC application
- Implementation of the security model (e.g. in short: *who can access what*)
- Implementation of password management
- Implementation of SSO and in general access control to the portal protected resources

Like all SSO systems, when a user tries to access a protected resource, he/she will be redirected to the login page, which will allow to access to the system (provided the user have been previously registered). All this workflow is completely independent on the CSNDC bespoke implementation and is controlled by EMSA at Enterprise level.

The only concepts that need to be specified for CSNDC are the following:

- Registration of user details in the CSNDC database
- Managing user roles

During the provisioning, one of the actions performed by the IdM is to invoke a Provisioning Web Service made available by the CSNDC, which as a result will store the user details into the CSNDC database. These user details are a combination of user registration details (which are needed by the IdM and also stored there, e.g. account, name, passwords, etc.), plus additional information which is required by the application for its specific functions. Typical example is the email to which oil spill alerts shall be distributed, which could be different from the standard user email with which the user was registered. The list of fields passed to CSNDC is specified in the CreateUser input data specification of [OIM-CSN].

One important aspect to highlight that a user can have multiple roles. Therefore, after a user was created, the user is given 1 or many roles (in order to be able to access the system, at least a user shall be associated to 1 role).

One important concept with regards to role management is that they will be mainly addressed using the Liferay page access management. In other words, a user will be able to access a given Liferay page, if that role is appropriately configured in Liferay (it has the rights grants to access that Liferay page). Please refer to www.liferay.com for instructions on how to do so. In order to cater for this, Liferay is configured with as many roles as are needed by each application (CSNDC in this case) and the provisioning workflow from the IdM, among the various operations, upon user creation, registers the same user to Liferay and associates it to the appropriate role. Therefore the user details and role will be somewhat shared by:

- OracleIdM
- Liferay
- CSNDC internal database

There will be some overlap in the information used by each component, but the final implementation of the user management is done by the combination of all of them. In short:

- Oracle IdM takes care of the access to the system and User provisioning/deprovisioning lifecycle
- Liferay takes care of coarse grained access to the system
- CSNDC takes care of fine grained access control and management of user details for the purposes of the CSNDC system itself

The list of user roles and their mapping against the coarse grained access to the CSNDC functions is represented in the following tables.

4.4.2 List of CSNDC users and roles

Liferay menu		Home	GIS Viewer	JReport	Planning	Alerting				Communication				
Liferaysub menu							Admin	Matrix	DAP		Wiki	Calendar	Forum	Document Library
UP20	csnalertconfig	x	x	x		x		x		x	x	x	x	x
UP05	csnauthofficer	x	x	x	x					x	x	x	x	x
UP06	csncsadmin	x	x	x						x	x	x	x	x
UP07	csncsoperational	x	x	x						x	x	x	x	x
UP10	csncsplanning	x	x	x	x					x	x	x	x	x
UP09	csncsuser	x	x	x						x	x	x	x	x
UP21	csndccommonservicedesk	x	x	x	x	x	x	x	x	x	x	x	x	x
UP01	csndcsysadmin	NOT USED												
UP04	csnfinancialofficer	x	x	x	x					x	x	x	x	x
UP15	csnguest	x	x	x						x	x	x	x	x
UP03	csnmssso	x	x	x		x	x	x		x	x	x	x	x
UP17	csnncaauthority	Created and managed by UP06 - Reduced set of attributes - At least one and only one per Country/Regional Agreement												
UP16	csnpublic	NOT USED												
UP02	csnservicedesk	x	x	x	x	x	x	x	x	x	x	x	x	x
UP14	csnsoadmin	x	x	x	x					x	x	x	x	x
UP12	csnsouser	x	x	x	x					x	x	x	x	x
UP13	csnspadmin	x	x	x	x					x	x	x	x	x
UP11	csnspuser	x	x	x	x					x	x	x	x	x
UP08	csnusergroup	NOT USED												

[illegible]

4.4.3 Implementation of the Data Access Policy (DAP) Management

One component of the User Management is the fine grained Data Access Policy management.

The design of the solution takes into account the following driving non-functional requirements:

- Minimise the coupling with the implementation of the CSN-DC, by implementing it in a modular component with centralised and well defined interfaces
- Optimise the performances of the data access system

The proposed approach should try to fit with the following concepts:

- Allow for a certain degree of expandability
- Focus on the core use cases, without trying to over-generalise the approach

The approach focusses on the management of systematic data, e.g. the data ingested systematically by the CSNDC. Data access rights shall be based on configurable rules.

The definition of the Data Access Rights can be represented by one or many combinations of the following elements:

- *User* (a user, identified uniquely by the ID, or collectively through the association to organisations, projects, etc.)
- *Action* (view, download, upload, edit, etc.)¹
- *Data Object* (the data to be protected in terms of data access)

So, for each of the macro use cases (see below), it is necessary to specify:

- How to identify a user
- What is the identification (and the granularity) of the Data Object
- What are the possible actions (depending on the use case and on the Data Object)

The concepts reported below are generally valid, possibly to be specified for each use case below.

User assignment

- Users belonging to one or more given projects
- Users belonging to one or more given organisations
- Users belonging to a given project of a given organisation

Authorised actions

1. View on the WUP
2. Download from WUP
 - High resolution (only applicable to EOP images)
 - Low resolution (only applicable to EOP images)

¹ Actions authorised for the users also depend on the user role (e.g. UP02, UP03, etc.). The role defines the actions allowed to the users in global sense (regardless of the data). On top of this there will (or may) be the configuration of the actions allowed for the individual data. Obviously the system will check both levels, e.g. in order to be able to download some data, the user has to belong to the role who has download capabilities and moreover has to be authorised for the specific data he/she is trying to download.

3. Add Feedback
4. Edit feedback

The systematically ingested data have the following characteristics:

- Are organised in services whereby a given service has a unique ID and is composed by 6 data packages
- The data packages which are subject to direct user access are EOP, OSN and DER, while the other are not directly accessible, but are only used to process/complement the information derived from the main data packages. However all data linked to services shall be tagged (i.e. including OSW, QNO, QUA).
- For EOP and OSN there is the possibility of visualising the information on the WUP and to download the data locally (single product download or standing order), while the DER can only be displayed on the WUP
- The packages have a clearly identified producer (the service provider who generated the data)
- Data packages have a clear time and geographic area definition
- For EMSA contracted services EMSA has the ownership (all packages). For third party services (e.g. national services) the owner is the third party.

Filtering criteria defined at ingestion time

As it is impossible to define all the possible criteria for defining the access rights to these data, we propose to implement an ingestion engine which will apply the data access rules at ingestion time and, consequently tag each incoming data sub-package into a *DA Repository*. A basic set of filters will be provided to define the data tagging rules, which initially are based on the following criteria:

- Geographic area
- Time window
- Data type (e.g. OSN, EOP, DER)
- For the EOP data also the platform (Envisat, Radarsat, etc.)
- Service provider

The concept will be similar to that of the standing order, e.g. a number of filtering criteria are defined and the result, instead of being a dissemination, as for the standing orders, will be a specific tagging of the data.

This approach creates a (number of) link(s) between user->action->data object that would allow to handle Data Access rights at later stage.

Authorised actions

Out of all possible authorised actions described in the previous section, here the possible actions for this use case are identified:

1. View on the WUP / Usage for alerting
2. Download from WUP
 - High resolution (only applicable to EOP images)
 - Low resolution (only applicable to EOP images)
3. Add Feedback
4. Edit feedback
- 5.

So, basically the application of a single data access rule, will be a combination of:

- Data identification criteria
- Authorised actions
- Users (authorised users may have different possible actions according to their role)

For example: all EOP packages falling within *AREA_1* and *TIME_WINDOW_1* shall be accessible in *view* and *download* mode by project *A/B/C* and in *view* mode (no download) by Organisation *A_1* and Organisation *B_1*. Whereby *AREA_1* is a geographic area (e.g. a polygon) and *TIME_WINDOW_1* is a given time window.

It shall be possible to define any arbitrary number of data access rules and they will all be enforced at the same time. As a result a given EOP package falling within the criteria of multiple rules, will be given access rights corresponding to the highest combination of the various rules.

So, every time a new data access rule will be configured according to the criteria defined above, the person responsible for this configuration (typically the EMSA administrator) will be prompted with the current list of projects/organisations, to which associate the appropriate privileges.

Each time a new package is ingested into the system, the access rules will be evaluated and, consequently, the data package is possibly tagged with the appropriate access rights. These access rights will be exploited by the system when checking whether to make visible and/or to authorise the download of a given package for a given user.

NOTES:

- The access rights is assigned to a given whole service including all data packages, so for example if there are multiple oil spills their visualisation will either be authorised or not authorised for all oil spills in bulk. Any geographic criteria will not be spill centric, but rather service centric.
- Access rights will be given at data ingestion time and cannot be changed at later stage (unless a direct update is performed on the appropriate database) for a given instance of a data package. EMSA will have the possibility to change the tagging matrix for special circumstances if necessary.
- A procedure will be provided to make existing data available to new projects (thus updating the tagging)
- Data access rights for a given rule can only be given to a single project and/or to a single organisation or to everyone. Different rights can be granted to each of these entities (e.g. Project A can view and download, Organisation A_1 can only view, everyone cannot view nor download). However each product can have several data access rights vectors (e.g. to be visible in different projects).

4.4.4 Implementation of the Operation-Based Data Access

On top of the DAP policy explained above, CSNDC also implements a data access based on the *Operation* concept. The term operation is mainly used in some EMSA integrated projects (e.g. IMDatE) to refer to a specific temporary adoption of data and assets for a given purpose. Examples of operation are:

- A campaign against illegal fishing

- Monitoring border areas for illegal immigration control
- Etc.

The basic concepts of operation-based access control are:

- Data must be associated to 1 or many operations.
- Association between data and operations is based on service ID (a service ID defines uniquely a EO scene and its associated packages as ordered from the POR, see § 4.8). All packages belonging to the same service ID will therefore inherit the tagging to operations
- Association is performed during planning and tasking (see § 4.8).
- A user will only be able to access (e.g. view, download, etc.) data if he/she belongs to at least one of the operations to which the service is associated.
- These rules are associated to both tasked data (already ordered but not delivered yet) and ingested and processed data (e.g. available through the GIS Viewer).

Here follow some more detailed specifications of the association between data and operations during and after planning:

- When starting a tasking session (see § 4.8.2) and the user creates a cart, it is possible to associate the whole cart to only 1 operation. This association is propagated to all scenes belonging to the current cart. This is also used in the JOU as a cost centre, to allocate the budget. This is called the *primary operation association*.
- After a tasking has been initiated, for each individual scenes it is possible to change the association to operations. In this case, through the POR GUI, the operator will be able to assign the individual scenes (one or a group of them) to multiple operations. This is called *secondary operation association*. The secondary operation association:
 - may or may not include the primary operation association
 - in terms of centre of cost has no effect (the JOU will still use the primary operation association for budget computation)
 - in terms of data access overrides completely the primary association
- Change of operation is not allowed when at least a user has confirmed the current service
- Change of operation is allowed after a scene has been delivered

For instance, if a given service was initially associated to operation A, and successively associated to operations B and C, a user belonging to operation A would not be able to access the data, but the cost will still be accounted to operation A.

The effect in terms of data access for the user is the same as already explained in §4.4.3 with some simplification:

- access control is only based on the operations (aka project) to which the user belongs
- access control does not differentiate the various types of actions that the user can perform on a given piece of data (e.g. view, download, feedback, etc.) but applies globally to all types of interactions
- access control is also enforced in the POR application
- access control is also enforced on the data in tasked status

4.5 PMA – Process Management

4.5.1 PMA Overview

The PMA is a fundamental component as it shall be able to **orchestrate various components inside and outside the CSN DC**. Here is a list of typical tasks that the PMA shall be able to orchestrate:

- Trigger processes/services or processing chains on the occurrence of the following events:
 - Arrival of a certain data type (data driven concept)
 - On user initiative
- Execute processing/services plugged inside the CSN DC
- Execute external hosted processes
- Trigger the ingestion of external data with various ingestion mechanisms, including triggering the execution of an external process which produces data that shall be ingested into the CSN DC
- Manage the in/out data flows versus the DAM, i.e. making data available to processing functions and storing the output data of the processing functions
- Manage the publication of the processing results on the WUP

The PMA shall be able to **combine various types of processing including systematic and on demand processing**. The processing may contain any combination of data analysis functions, including pre-processing, internal/external processing and definition of the data access policies. The processes of ingestion and publication on the WUP are managed by other components, typically via the DAM. By combining the IIF, the DAM ingestion functions, the concatenation with the PMA and the WUP, it will be possible to create complex end-to-end workflows that satisfy all the requirements for the CSN-DC. The PMA component is mainly focussed on the processing part, which can be composed on any sequence of processing sub-blocks of internal/external modules.

The PMA component is based on the Task Manager, based on the ACS COTS *PDS Thin Layer*

This has the following characteristics:

- implements a SOA paradigm
- easily configurable
- based on robust well tested ACS COTS

The Thin Layer strong points are:

- implements distributed processing enforcing processor decoupling and load balancing
- implement complex data correlation rules, with combination of multiple data types and correlation strategies
- efficiently implements the data driven triggering

Therefore the **Thin Layer is particularly useful when implementing systematic processes** which are routinely executed when images are made available, especially when complex data correlation rules shall be put in place. This may be the case of systematic geometric data processing or systematic oil spill analysis workflows.

The Thin Layer will be mainly configured initially with the capability of routinely executing a certain number of systematic processing tasks. Whenever additional types of systematic processing tasks,

defined during the lifecycle of the CSN DC operations, become consolidated, they can be added to the Thin Layer task list, with a simple configuration operation.

4.5.2 PDS Thin Layer

The **PDS Thin Layer** (with the addition of a small wrapper, which creates the *Processing Orders*, see below Figure 4-16) implements the whole PMA component. The Thin Layer strong points can be summarised as follows:

- Capability of **implementing a SOA-like paradigm**: processes may be distributed on an environment that includes many processing resources
- **Decoupling of processing resources from the controlling infrastructure**: the infrastructure is unaware of the implementation of the processors, likewise the processors are unaware of the implementation of the infrastructure
- **Implementation of standard and simple file-based interfaces**: processors interfaces are mainly based on XML file exchange
- **Load balancing and scalability**: the same processing function can be deployed on multiple CPUs and the Thin Layer automatically distributes the processing on the less loaded ones
- **Activation of a processing in NRT** i.e. when all mandatory input becomes available. This feature is very important for systems like CSN DC, having services requiring it.

These concepts are well illustrated in the figure below, showing how **the processors can easily be deployed on a distributed processing environment** and be coordinated by the CSN DC Infrastructure. Processes may be deployed on any series of workstations, potentially inside or outside the CSN DC perimeter. On each workstation the processing facility is composed by one instance of the Thin Layer and by one or several hosted process each with its own configuration file.

4.5.3 KEO

KEO System implements a distributed component based programming and processing environment (Algorithmic Image Processing – AIP), supporting extraction of information from EO images by permitting to:

- easily and visually create Processing Components (called FEPs – Feature Extraction Processors), also from interactive training (no code writing),
- semantically catalogue Processing Components,
- semantically discover Processing Components suitable for a specific task,
- start systematic or occasional execution of Processing Components against EO images to extract information.

KEO is not used in the standard processing workflow of the CSNDC. It was delivered as a separate tool, that can be executed manually, e.g. for testing new processors, etc..

NOTE: the term “Feature Extraction Processor” is used in the KEO system, since KEO was conceived with the functional purpose to orchestrate mainly image feature extraction algorithms. Although we will continue using the same term in this context, in order to clarify that exactly the same component of KEO is used in the CSN-DC, it is worth highlighting that the FEP can in fact encapsulate any type of processor (e.g. data processing, querying data, retrieval, etc.). So the term

“Feature Extraction Processor” shall be intended to be equivalent to a more generic term “Processor/Service”.

The system is able to orchestrate modules of different kinds, such as Web Services, CLI (command line interfaces) programs and Java-coded extensions, with a variable level of complexity. From this point of view, the system can be perceived as an open container hosting pre-defined modular components, but also able to enlarge the set of provided functionalities by registering and later managing new modules coming from the external world. Hence, KEO can create, edit and manage processing flows and/or components working on data to and from external interfaces. This concept allows us to specify a *Functional View* of the system mainly directed to those software tools, like the Feature Extraction Processor engine, which perceive KEO as workbench for the assemblage and the operative management of new or better business processes starting from simpler, atomic, well-defined and modular functionalities.

4.5.3.1 KAFE Servers

The KEO Actuator and FEP Engine (KAFE) servers are the back end of KEO FEP functionalities. They are based on the J2EE 1.5 technology, the last Java Platform Enterprise Edition. The KAOS Client Application interacts with the KAFE servers in order to manage image-processing modules and connect them into complex graphs. The KAFE server is made by two web applications:

- the KAFE engine, that is the front end to a network of computers (the actuators) addicted to the execution of brick modules;
- the KAFE actuator, a web application deployed on each machine of the network.

The main engine's function is to be the access point to the computation network, choosing the best actuator on which the process will be executed and managing data types used by modules. The engine contains a list of available actuators and each request submitted by a client will be redirected to the appropriate actuator by the engine.

The main actuator's function is to store, manage and execute brick modules. Engine and actuators have a MySQL database – that for simplicity have the same schema – in which are stored information about data types, modules and processes status. This MySQL database is used locally by the engine and by the actuators only.

KAFE servers (engine and actuators) are based on the J2EE 1.5 technology, such as the KARISMA server, sharing the same architecture and structure.

The KAFE servers could run on any application server providing support for J2EE 1.5 (Sun's Java Application Server, JBoss application server, Oracle application server, Glassfish). In the CSN-DC implementation Glassfish will be used due to the huge support that gives for the latest Java EE 5 technologies and because it is supported by NetBeans IDE ver. 6.0 or greater.

The servers are mainly made by:

- a set of actions (java classes) to perform the operations needed, using session beans, that are facades around the entity classes (emulators of database tables);
- a Context Listener that, on load of the application, instantiates actions and make them available, putting them into the servlet context;
- a MainServlet (the front controller of the application) whose job is retrieving the right action from the context and forward the request to it.

Moreover the Context Listener of the actuators performs a set of initialization operation, as registering to the engine, cleaning the database and downloading from db into a directory the jar used by java modules, pinging periodically the engine.

4.5.3.2 Feature Extraction Processor

The KEO system allows users to create and add to the systematic ingestion and processing chain a custom Feature Extraction Processor (FEP) obtained patching together a set of processing modules. These modules can be chosen from:

- the set of primitive modules, performing simple operations on simple data type values (i.e. adding two integer numbers, regular expression processing of strings, ...);
- the set of factory modules, acting on complex data type values (i.e. image format conversion, image warping into a new coordinate system, ...);
- the set of user-provided modules, performing custom feature extraction algorithms on any type of data values;

Those modules are atomic, in the sense they are independent programs, Web Services or pieces of Java code, thus constituting the fundamental bricks of the system, while FEP modules are made by chains built-up of atomic pieces and possibly other FEPs. A FEP module must satisfy specific rules and interfaces as specified in details in the FEP Interface Control Document. Non-primitive processing modules are ingested into the system (by ACS KEO Team personnel for factory modules and by any authorized user for user-provided ones) by means of a tool that allows the definition and classification of the module, and eventually the uploading of the corresponding processing code.

4.5.3.3 Data-Flow Explained

A data-flow consists in a directed graph of processing nodes connected with FIFO data queues. This pipelined architecture lets applications be built from small-size reusable components stitched together with queues. Since it's a pipelined architecture, it naturally takes advantage of many processing units (computers, CPUs, cores, etc.).

Data-flow is an alternative to the standard von Neumann model of computation. Typically, we think of a program as a series of instructions each executed one after the other by a processor keeping track of its progress with an instruction pointer. In data-flow, on the other hand, channels transmitting data in one direction join computations to one another. Conceptually, you can think of this structure as a directed graph with data channels as edges and processes performing computation on the data as nodes. The processes each operate only when data is available—the data flowing through the network is all that's needed to organize the computation.

The immediate advantage is that many of the processes can be operating simultaneously, thus allowing dataflow applications to take advantage of multiple hardware nodes and hardware with multiple CPUs or processor cores. Notice that concurrency happens externally to the process. The developer doesn't have to bother with threads, deadlock detection, starvation, or concurrent memory access to build parallelism into his application.

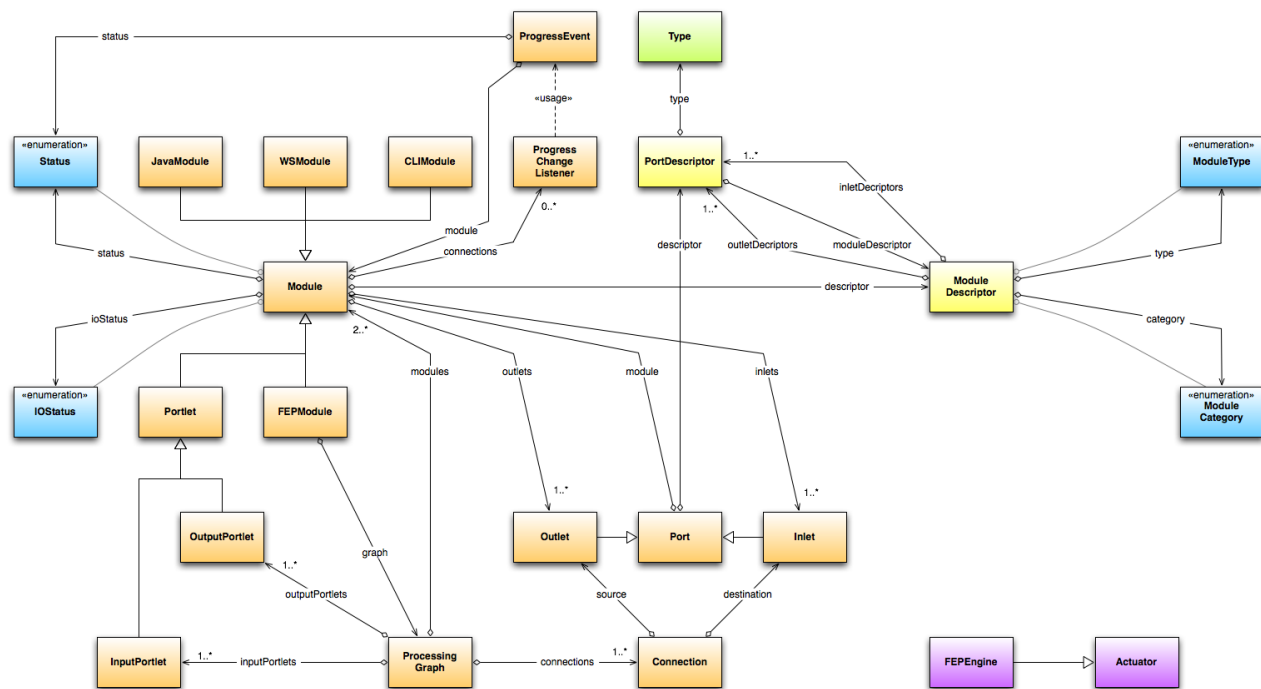
This type of implicit parallelism stands in stark contrast to the concurrency mechanisms of many other programming paradigms. Moreover, data-flow allows composability: as long as the I/O contract is correct, sub-graphs may replace nodes or be spliced between them in the original data-flow network. This facilitates both program correctness, since sub-graphs can be tested as they're constructed then linked together to form larger programs, and code reuse, since commonly used sub-graphs can be copied from one application to the next.

The outcome of a data-flow network is determined uniquely by its input, regardless of the order in which processes fire. Firing order impacts queue sizes and performance, of course, but this can be dealt with elsewhere besides explicitly within the program itself, dramatically simplifying the task of the dataflow programmer.

4.5.3.4 Architectural Overview

The FEP system is basically made of three parts:

- A `FEPModule` contains the `ProcessingGraph` describing its internals in terms of `Modules` and `Connections` (the nodes and edges of the processing graph). As you can see from the figures, a processing graph must be made of at least two modules (an input and an output portlets) and one connection (linking the input with the output portlets, thus realizing an identity module).



Each Module has (for what said before) at least one Inlet and one Outlet Ports. Common properties of both Modules and Ports are described in the corresponding ModuleDescriptor and PortDescriptor objects, and will be displayed and manipulated in the FEP Designer window.

The Feature Extraction Processor (FEP) Engine is a data-flow engine that executes and coordinates Feature Extraction Processors made of factory processing modules and other FEPs distributed in a Local Area Network (LAN). FEP Actuators are the services that will make available FEP platform-dependent programs (or modules) to the engine. Actuators will be deployed on virtual machines

dedicated on specific KEO processing nodes, containing run time environments for specific applications (e.g. Matlab, Definiens).

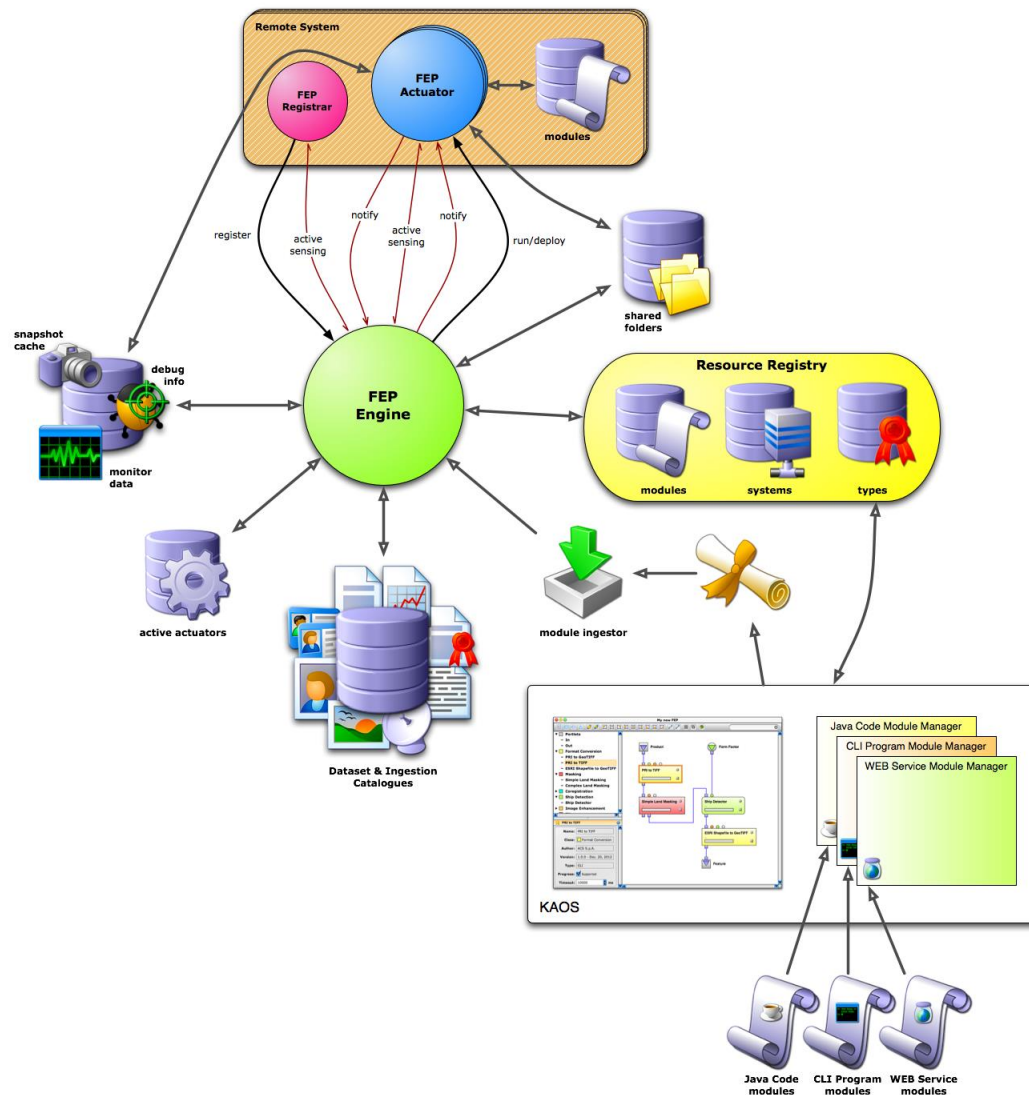


Figure 4-10: KAFE Engine/Actuators architecture.

The FEP Engine keeps in its local storage a list of all available modules addressed by their identifiers (FIDs, see also Figure 4-9). Moreover it keeps also a list of all registered actuators that will physically run modules (Java methods, CLI process or Web Service functions). These lists will then be used to dispatch processing graph execution to the suitable actuators. When a remote system is started and become available to the KEO local network, its FEP Registrar will start looking for the FEP Engine (see Figure 4-10).

Please note that in the KEO jargon the term “remote system” refers to a system which is external to the FEP engine, regardless of its physical deployment. In the CSN-DC context these systems will still be inside the CSN-DC network, and are not physically “remote”.

When found, the local (relative to the remote system) list of available modules is transferred to the FEP Engines that will store it in the storage's lists of available systems and modules. Moreover, the

communication between engine and actuators is bidirectional: periodically one system calls the other (through an `isActive` method) to see if it is still available. If no answer is received, the partner system is considered unreachable, a situation that could trigger appropriate fall out procedures (for example: removing of the corresponding modules from the storage and trying to reallocate execution of pending modules on a different system).

When a FEP module must be executed the corresponding module description is retrieved based on its FID (the FEP identifier), then:

- the description (saved in a proprietary XML format) is parsed to discover the modules it is made by;
- for each module (or node) in the processing graph a corresponding actuator is activated in the appropriate remote system, and relative information is stored in the database;
- active sensing is started in both the actuators and the engine;
- for each connection (or edge) in the processing graph the target actuator is registered to the source one as listener of data availability events (moreover if some FEP designer windows are monitoring/debugging the FEP module, then they will be registered too, in order to graphically highlight connection when data availability will be notified);
- all activated actuators are started.
- All data exchanged between modules is stored into a snapshot cache database in order to:
- be easily and uniformly exchanged between modules (massive data is exchanged by means of shared folders already defined in KEO as working area);
- Provide a starting point for failure recovery procedures triggered by the active sensing ones when a systems or network breakdown is discovered.

The same database and data (plus some more) will be used also for FEP modules monitoring and debugging. The FEP Engine will be probably implemented as a web application with eventually some functionality exposed to the outer world by means of a wrapping Web Service (like KARISMA).

Actuators and registrars will never be accessed directly but only through the engine: also the ingestion of primitive, CLI and Web Service modules will be mediated by the FEP Engine (through the ingestor module).

4.5.3.6 FEP Actuator

FEP Actuators will make platform-dependent modules (Java coded, CLI programs, Web Services) available to the engine. An actuator is mainly composed by three threads (see Figure 4-11):

- the Data Listener, whose purpose is to continuously listen for inlets data coming from source data provider;
- the Execution thread, that will execute the module's processing algorithm (invoking corresponding Java code or Web Service function, or starting a CLI program). It is normally sleeping and awoken when all inlets have compatible data;
- the Data Producer, that will notify registered listener when outlets data is produced. It is normally sleeping and awoken as soon as new data is produced in any module's outlet.

Data queues are backed up in the snapshot cache database and (for massive data) in the KEO shared folder(s). Threads will save running status in the monitor data and debug info database.

Actuators have a local database containing the list of modules that can be run (see Figure 4-10).

Modules are ingested in the FEP engine systems (see [4.5.3.7]) and loaded in all actuators supporting the modules type and platform. The communication between actuators and the FEP engine occurs via the HTTP protocol on the standard ports (e.g. 8080).

4.5.3.7 FEP Module Ingestion

Modules (primitive, factory and user-provided) need to be described and the corresponding processing code (Java, CLI program or Web Service) must be ingested into the system, in order to

be deployed on all machines matching the required platform system requirements. This will be done using the KAOS client application's FEP Module Manager dialog (see Figure 4-12).

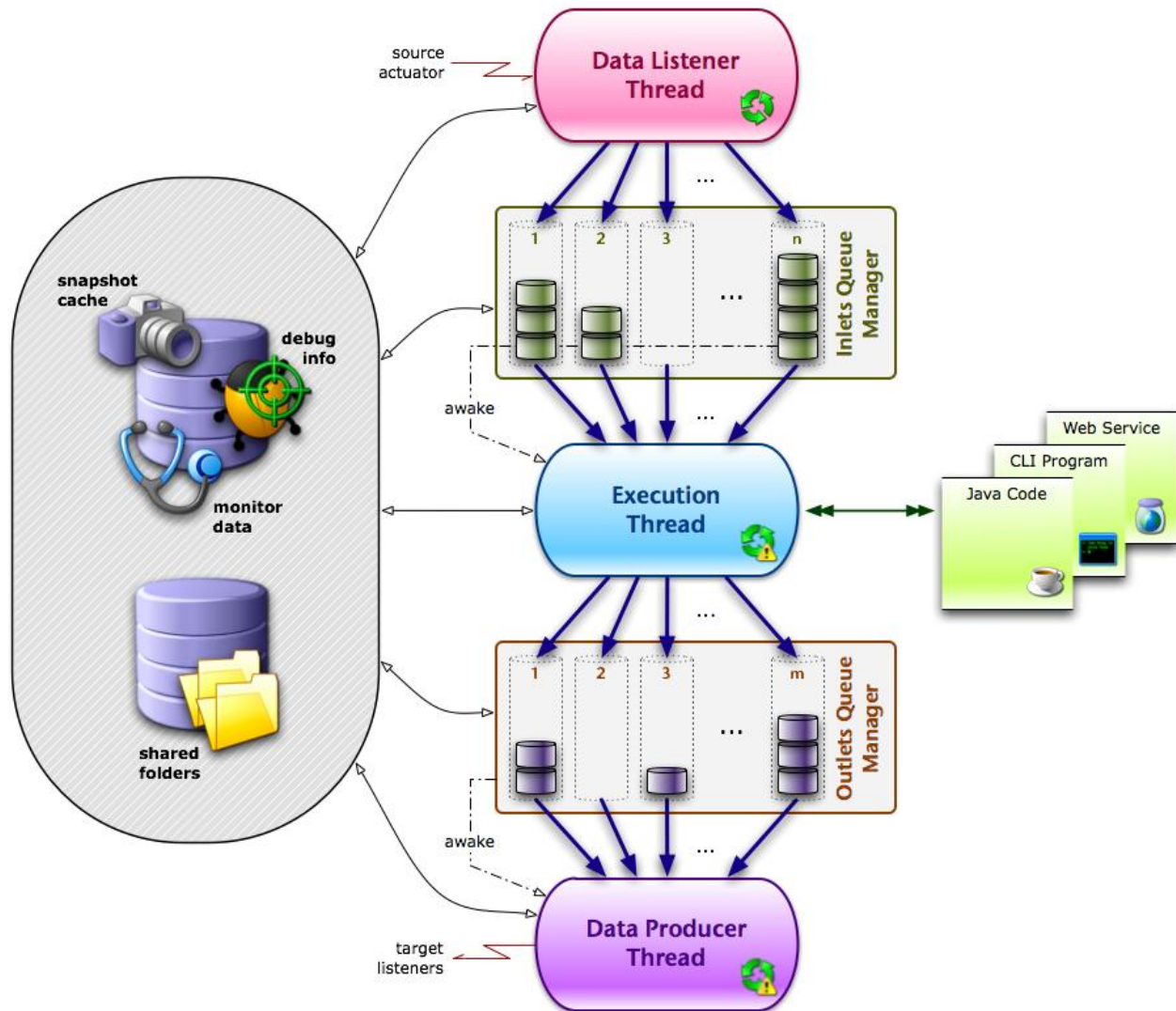


Figure 4-11: FEP actuators architecture.

The main area of this dialog is for a table showing the available modules. The table can be updated, reloading information from the FEP Engine, pressing the **Refresh** button anytime.

For each module in the table the following attributes are shown:

- the category (column **C**, an icon indicating if the module is *Primitive*, *Factory* or *User Provided*);
- the running status (column **R**, where a check mark indicates top-level modules running or scheduled for);
- the SSE published flag (column **S**);
- the name (column **Name**);
- the type (column **Type**, possible values are: Java, CLI, WS and FEP);
- the class (column **Class**);
- the platform where CLI modules will run (column **Platform**).

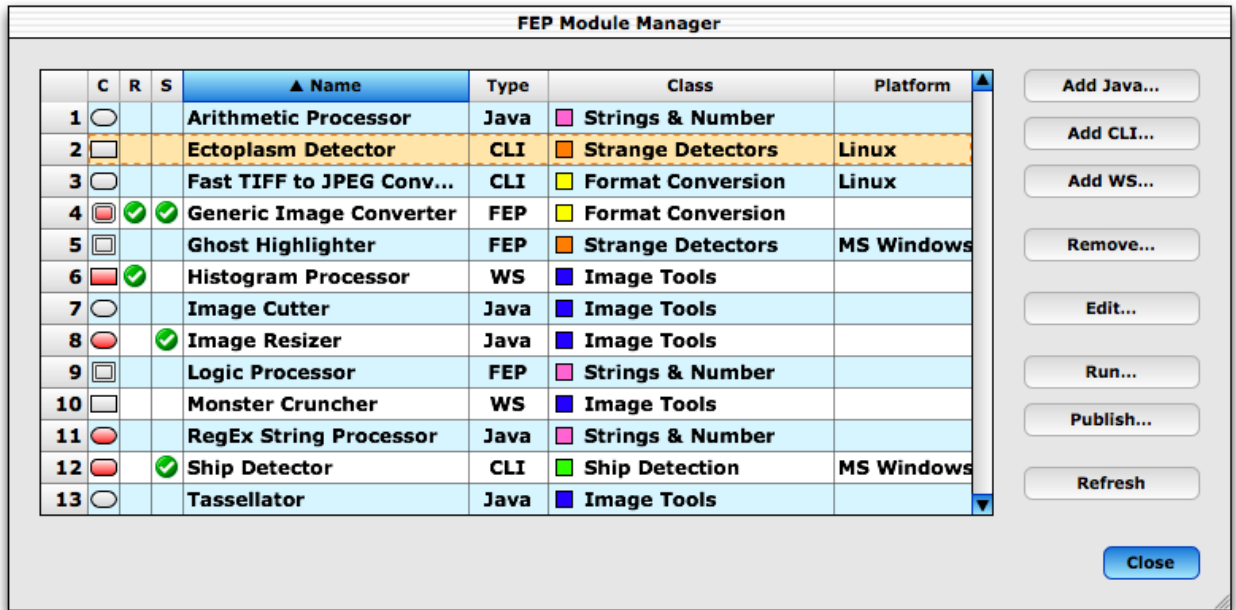


Figure 4-12: FEP Module Manager dialog.

The three buttons **Add Java...**, **Add CLI...** and **Add WS...** allow to ingest a new Java coded, CLI program and Web Service module, as detailed in [KEO-TN].

The **Remove...** button removes the selected module(s) from the FEP engine and from all actuators having a copy of it. This operation is dangerous and a confirmation dialog will be prompted to the user for a confirmation. Being a module possibly used inside other FEP modules, the confirmation dialog will have a button allowing for a complete dependency search in the FEP engine in order to understand if the chosen module can be safely removed or not.

It is also possible to edit (through the **Edit...** button) existing modules in order to upgrade some information and the implementation code. The **Run...** button, enabled only when a top-level module is selected, allows to schedule the selected module for execution.

4.5.3.8 FEP Designer

For the KAOS application a Feature Extraction Processor (FEP) Designer component has been developed in order to allow users to define processing modules to be executed by the FEP Engine using a visual programming. Every designed module can be used in other processing graphs or as a Feature Extraction Processor in the FEP Driven Ingestion when the required rules are satisfied. The FEP Designer is one of the KAOS MDI (Multi-Document Interface) windows, and it is sketched in Figure 4-13. It can be opened by means of the File->New->FEP Design or File->Open->FEP Design... menu commands. More than one designer window can be opened at the same time, each showing different processing modules.

The designer's area is composed by:

- the sketchpad (filling the most of available space), where the user programs (i.e. draws) and debugs its processing module;
- the list of available modules organized by class;
- the inspection panel, showing info and parameters about the selected module (see Figure 4-13) or the selected inlet or outlet;
- the tool-bar with all possible actions on the topmost window's area.

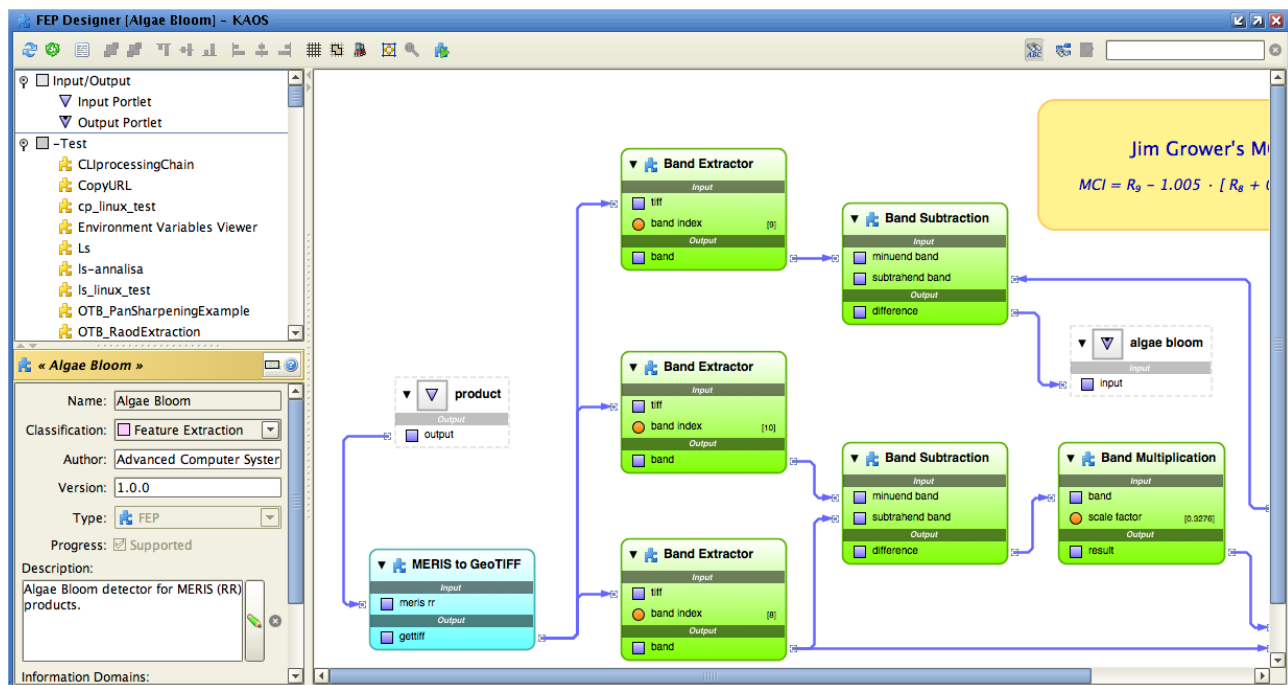


Figure 4-13: FEP designer window.

A FEP graph is made interconnecting processing modules: modules are selected from the modules list in the left side of the designer and dragged into the sketchpad. Then a connection from a source module is started dragging from one of its outlets and terminated into an inlet of the target module. Other kind of objects can be inserted in a module (non-processing, simple graphical shapes and text boxes) for documentation purposes or to make the graph more readable. Diagrams can be easily and quickly laid out thanks to a grid that can be made visible or not, and whose intersection points objects and connections can be snapped on. Grid visibility and snapping are independent of each other and can be set using the corresponding buttons in the tool-bar.

In the FEP Designer a module is graphically represented by a pseudo-rectangular shape that can be directly manipulated with the mouse, and connected with other modules thus realizing the processing graph.

Three categories of modules exist (primitive, factory and user-provided, and for the last two atomic and FEP variants exist), each one represented with different graphical shapes (see Figure 4-14). A module is composed by a body (in one of its three possible shapes), a set of inlets on the upper bound, and a set of outlets in the lower bound (see Figure 4-15).

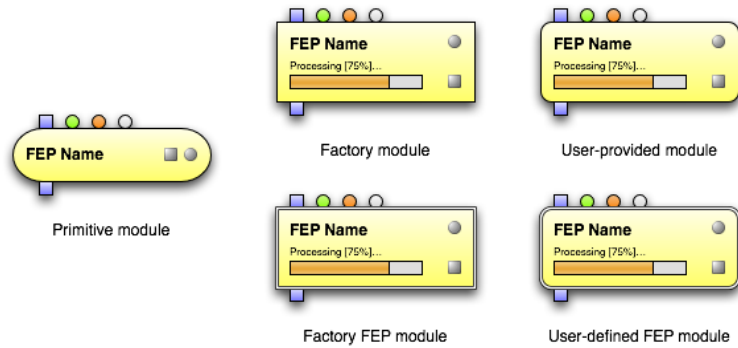


Figure 4-14: Processing module shapes.

Inlets are the target pads of a connection and are related to input parameters of the underlying processing module, while outlets are the source pads and are related to the outputs of the module. This arrangement will produce a visual flow of data from the top of a diagram towards its bottom. Outlets have always the square shape and the [102; 102; 255] colour ("orchid"-like). Inlets instead can have different shapes and colours.

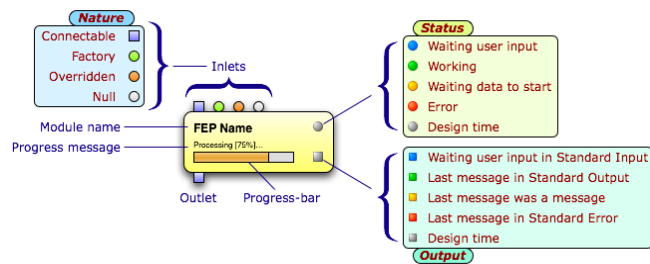


Figure 4-15: Anatomy of a processing module.

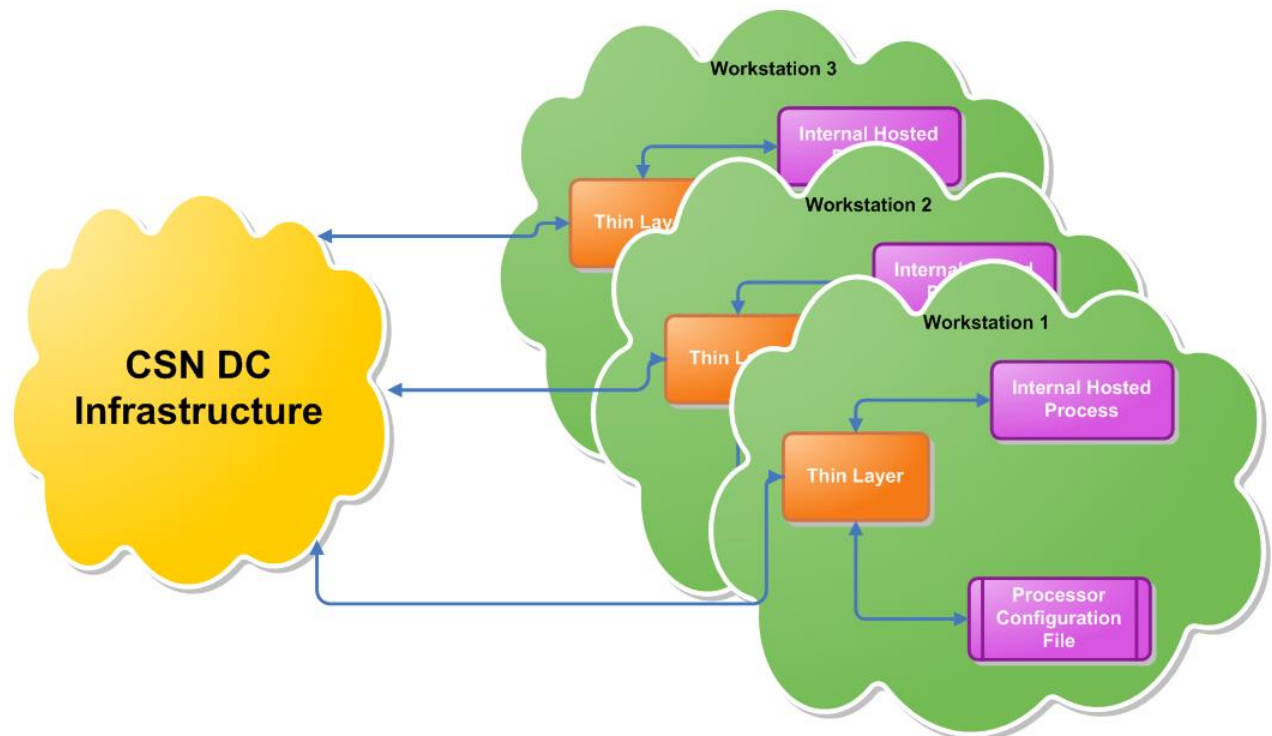


Figure 4-16 - Thin Layer architecture on a distributed environment

Within each workstation a Thin Layer interfaces all of the processors (one or more) residing on that workstation. They are represented in the figure above by *Internal Hosted Process*. In fact this mechanism will easily allow to also execute external hosted processes, so long as they can be wrapped into a web service call executed by the internal hosted processors.

The Thin-Layer implements the following functions:

- polls the *Processing Order* queue,
- retrieves the data needed for the processing from the DAM
- runs and controls the execution of the processors.

As far as the processors are concerned, they are simple executables launched by the Thin Layer. One processor could consist of one or more executable. The mechanisms for the definition of a generic processor and for its calling through the Thin Layer agent are outlined hereafter.

All processors are expected, from the architectural standpoint, to be built as a series of stand-alone executables (tasks).

The exact list of parameters that shall serve to the processor characterisation is provided in the Task Table interface described in the document [PDS-ICD]. No other parameter shall be necessary to correctly define the processor behaviour. Several processors may reside in each one of the workstations together with a Thin Layer agent. The Thin Layer knows the characteristics of a generic processor at the workstation level through two XML configuration files:

- Workstation (WS) Configuration file
- Processor Task Table

The **Workstation Configuration file** contains all the information needed by the Thin-Layer to identify what kind of orders can be satisfied on the processing workstation. The Workstation Configuration file identifies which ones are available and what orders they are enabled to carry out.

The **Processor Task Table** contains the definition of the processor, i.e. the number of composing executables and list of input, output and intermediate file types for each of them. A processor Task Table is defined for each SW processor version. Through this information the Thin Layer can download all the inputs needed by the processor (specifically by each of the tasks composing the processor) from the DAM. At Task Table level it is also possible to specify if a file is mandatory for the processor to run.

4.5.3.9 The PROCESSOR calling mechanism

This mechanism for calling a processor is illustrated in

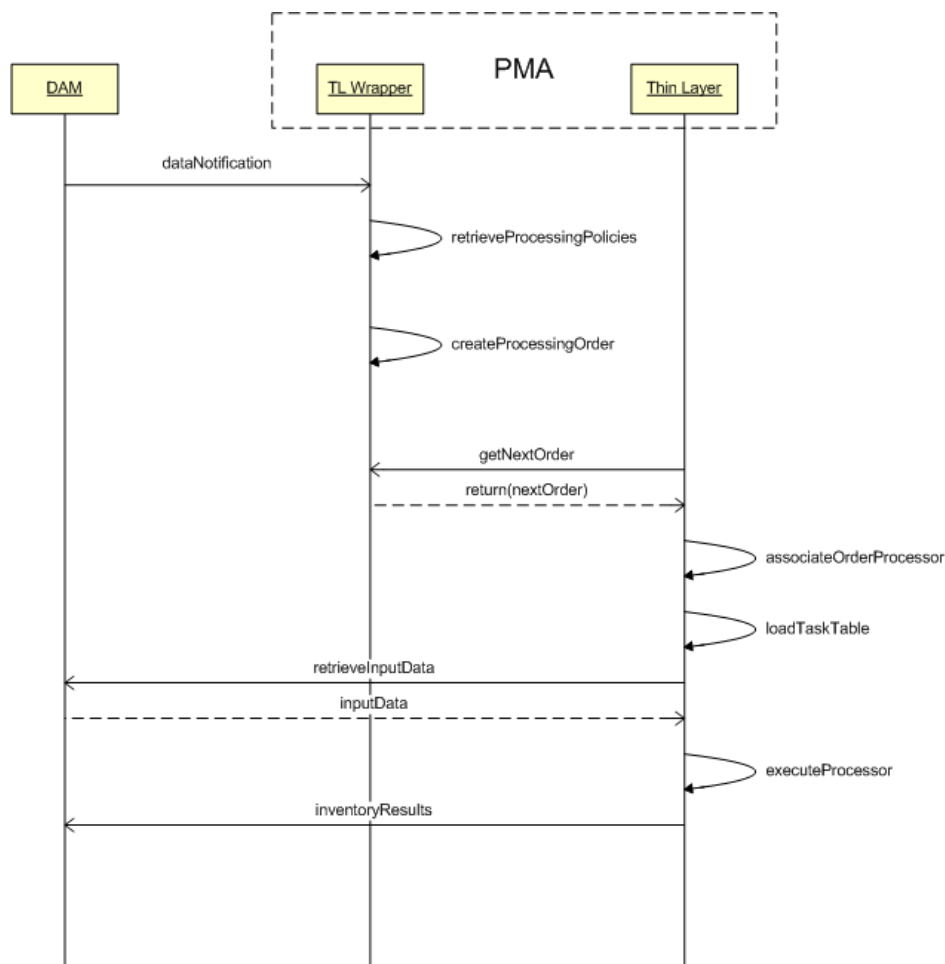


Figure 4-17 (although sequence diagrams and in general dynamic behaviour of the system are included in the next chapter, this sequence diagram has been included here, as it supports explanation of the activation mechanism).

In order to better explain this mechanism it is necessary to introduce an additional PMA sub-component, called the *Thin Layer Wrapper*, which is responsible of creating the connection between the arrival of data and the actual activation of the Thin Layer. It is also part of the PMA, but differently from the Thin Layer, the Thin Layer Wrapper is not deployed on the workstations where the processors are installed, but is centralised in the CSN-DC system. It polls the DAM inventory table every time new data arrive and based on a pre-defined configuration, creates Processing Orders. The overall mechanism is explained hereafter:

- upon start-up, the Thin Layer loads the Workstation Configuration file and goes quiescent
- when a file is ingested the DAM sends a data notification to the PMA (see also Figure 5-1).
- on the basis of a file type and of a configurable processing order policy, the PMA (namely the Thin Layer Wrapper) creates the so-called *Processing Order*: the Processing Order is physically implemented as an entry into a table (called the order queue), which defines the following major parameters:
 - processor to be executed
 - time range of the processing
 - main file type (typically the file type that has triggered the processing)
- based on the parameters above it is possible to identify which processor shall be triggered and on which file type and time range
- the Thin Layer checks the order queue and maps the existing orders with the configured processors (the list of processors are in the Workstation Configuration File)
- If one such processor is found, its Task Table is loaded, the order is pre-booked and the DAM is queried to check if all needed input files are available.

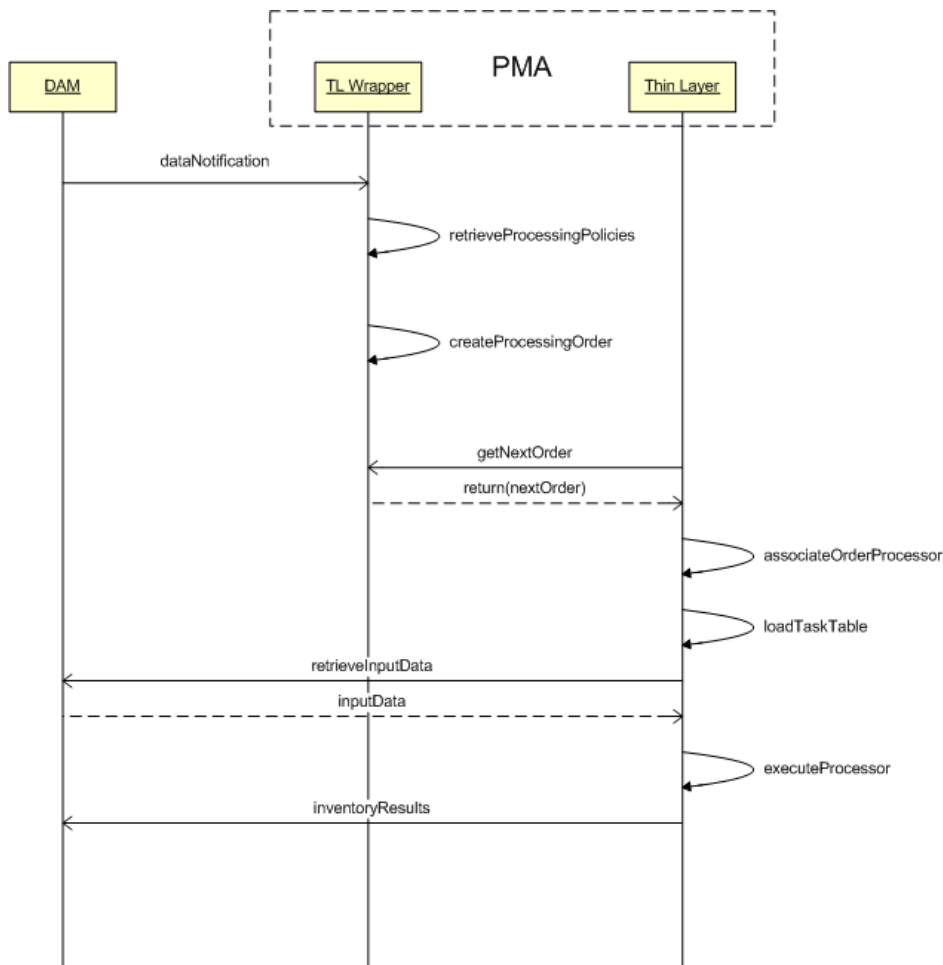


Figure 4-17 Thin Layer activation mechanism

If all input data are available, then the processor is executed. In particular the following actions are undertaken:

- the order is booked for execution in the queue
- the input files are retrieved and pushed onto the processing working directory. A **Job Order** file containing the list of all input files required by each task of the processor and expected output files by each task of the processor is created.
- the tasks composing the processor are run in the exact sequence, passing as unique argument in the command line the Job Order, and their execution monitored
- upon accomplishing the processing task the Thin Layer collects output products and stores them into the DAM with accompanying inventory data.

More details about this mechanism and the details of the interfaces are explained in [PDS-ICD].

This mechanism implements a real **data driven approach**, rather than an event driven approach. The Thin Layer is aware of a series of Processors which are *subscribed* to it and, based on the processor configuration file (aka the processor Task Table, see [PDS-ICD]), knows what are the data types necessary for triggering the processing and which are the necessary correlation rules for combining multiple different data types before the execution of a certain processing. This **allows the Thin Layer to know exactly when all necessary data for a certain processor are available in the system and consequently trigger in Near Real Time the execution of that processing**.

The capability of **coordinating the input data collection** is one of the most powerful features of the Thin Layer. Typically a processing function is based on the combination of different file types, such as EO raw data, precision orbit files, auxiliary files (e.g. processor configuration, external calibration parameters, etc.) ancillary thematic layers, etc. Some of these file are **mandatory**, whereas others may be **optional**. The association between files of different types may depend on a combination of rules, which could be based on time correlation (files shall belong to the same time segment), geographic correlation (files shall belong to the same geographic area) and other complex rules (e.g. select the available orbit file with the highest precision available).

The Thin Layer offers a simple configurable mechanism for exploiting these data correlation rules, based on:

- An expandable library of rule types
- A configuration file which defines which rule types to use and with which parameters

Typical examples of frequently used rule types are: time overlaps, geographic overlap, file sequence merging, etc. These rule types may have different configurations. For example, when defining geographic overlap between 2 file types (e.g. SAR file and its relative vessel information) it is necessary to configure if a partial overlap can be accepted or a total overlap is mandatory, what to do in case of a partial overlap, etc. As indicated above the library of rule types can be simply expanded by slotting in plug-in functions that can be implemented in any programming language, including scripting. This modularity allows to cope with any data input preparation complexity, thus enhancing the expandability of the solution.

To give a **concrete example**, this data driven paradigm well fits with the need of implementing a workflow to automatically perform geometric correction, radiometric normalisation and speckle filtering on images, which requires the following data types:

- The image to be geometrically corrected
- The SP file indicating the geometric displacement to be applied on the image

Every time a SAR image is retrieved the Thin layer will automatically retrieve the SP file as part of the Image quality notification package. SAR image and the quality notification belong to the same processing package, they arrive into the CSN-DC asynchronously. The Thin layer will identify the corresponding files matching and prepare them for the processing activity.

Although the above discussion is focussed on systematic processing, the Thin Layer allows also to execute ad-hoc processes. The underlying activation mechanism is exactly the same, with the simplification that the Processing Order is not generated by the Thin Layer wrapper, but generated immediately upon user request (e.g. the user clicks on the GUI and the system automatically generates the Processing Order corresponding to the function activated by the user, so as to trigger the Thin Layer execution).

4.5.3.10 Configurability and usage of the Thin Layer inside the CSN-DC PMA.

The Thin Layer can be configured using a user-friendly GUI, which will allow to set up the list of processors to be managed, the configuration of each processor and the input preparation/correlation strategy, i.e. the rules based on which a certain routine processing can be triggered, i.e. to create the Processing Order.

Processors can be developed using any technology independent on the implementation of the CSN infrastructure and all interfaces, including configuration and input data are performed via files (namely an XML file for configuration and working directory for input/output files management, i.e. the Task Table, cfr [PDS-ICD]).

Processors can wrap up any type of processing, including call to internal or external web services, call to external functions for activating external oil spill models, retrieval of external auxiliary information for complementing the analysis of the scene information obtained from the SPs (e.g. collection of vessel traffic data data, query to external systems, etc.).

The Thin Layer will be used as a baseline the backbone for the implementation of all services required for the systematic processing in the CSN-DC, leveraging its NRT capabilities.

4.6 WUP – Web User Portal

The Web User Portal is the central user interface for accessing data, information, alert and messages produced by CSN DC. Moreover, a number of operations of the other components like COM, POR and PDE are made available through the WUP. Next figure reports high-level WUP architecture

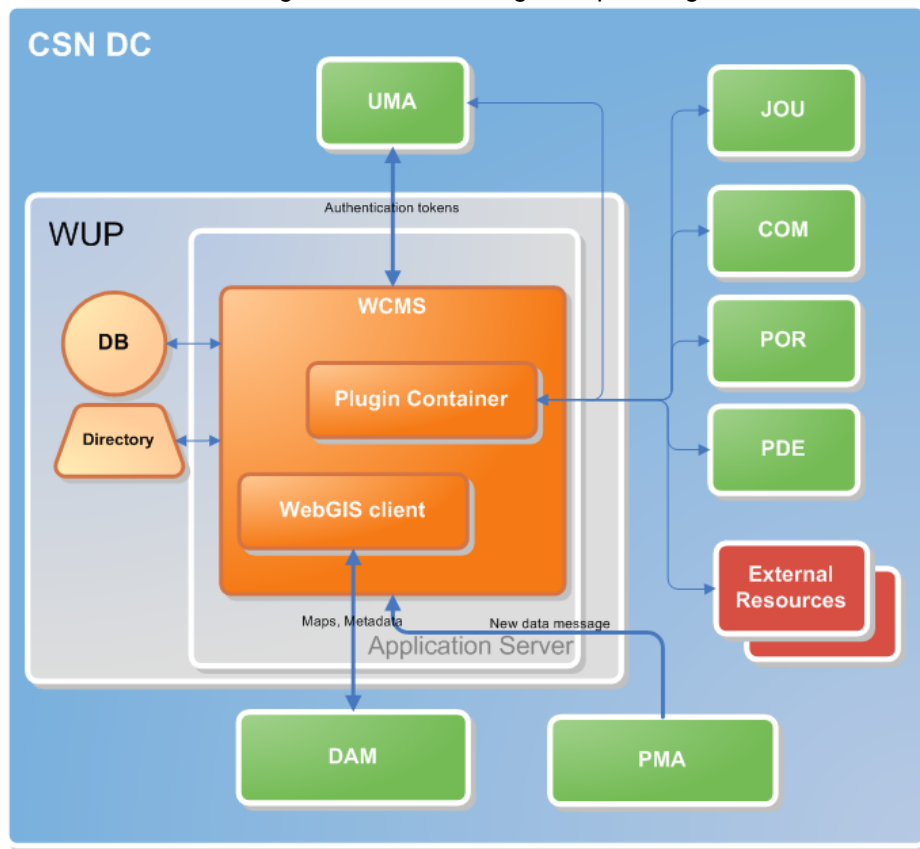


Figure 4-18 WUP context and main interfaces

The WUP component is largely based on OGC WMS/WFS/CSW clients that manages data requests to a spatial-enabled service and make data accessible through http protocol. Data and metadata are served by DAM that acts as the data service provider of the WUP geo-enabled portal.

The WUP exchanges data (raster maps, GML vector data representing oil spills, vessels and EO scenes, etc...) with DAM through a set of OGC services.

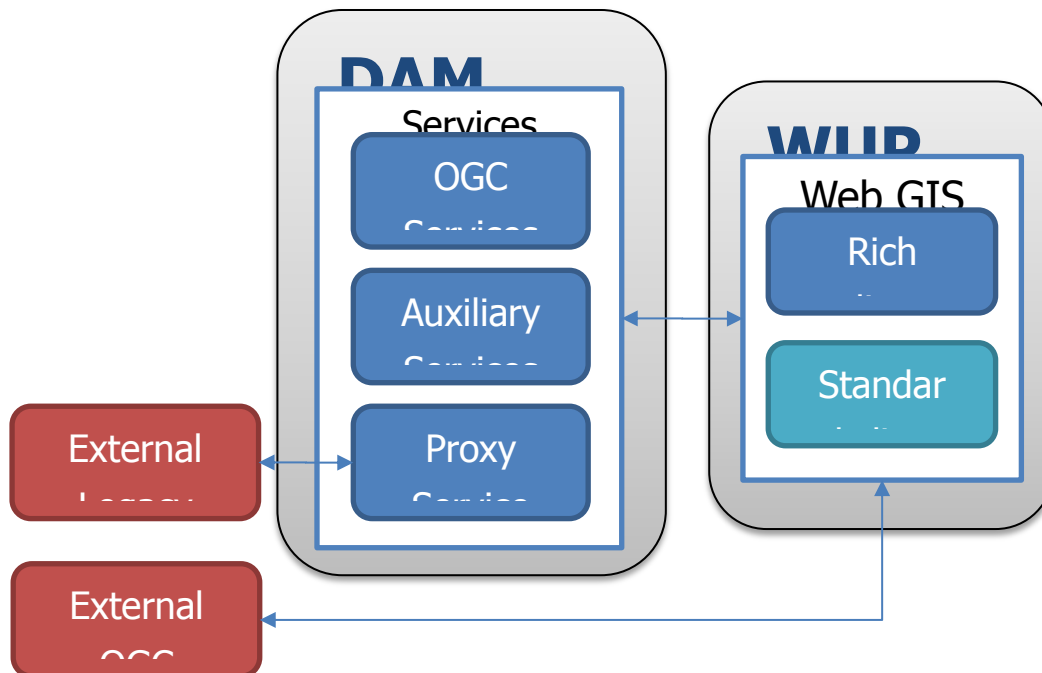


Figure 4-19 WUP as a DAM's services client

In addition to the standard OGC services, for optimisation purposes the WUP interacts directly with the database for its own specific business functionalities. The reason why the internal connection to the database are performed using direct connection (and not standard services) are manifold:

- The business logic of the WUP is quite complex in terms of data interlink, due to the following:
 - Need to retrieve in a single data request a number of different interrelated data objects
 - Need to navigate among data object following a very specific navigation paradigm
- Performance of direct query (e.g. ODBC connection) are always better and more controlled than performance using a standard mechanism (which for example requires marshalling and unmarshalling of potentially very long XML files)
- Being an internal interface there is no constraint to adopt a standard solution

The high level data model of the objects managed by the WUP can be summarised as follows. It summarises the following main data objects:

- EOP: the data regarding the Earth Observation products
- Oil Spill: data regarding the oil spills and associated information
- Detected Ships: data regarding the detected vessels
- AIS data: AIS vessel data retrieved from the IMDatE system
- Feedback: information about the feedbacks inserted by the users
- Alert: info about the alerts generated and the specific configuration of each coastal state
- Planning: data regarding planned and tasked scenes
- User DB: information about the CSNDC users and their relationships with the alerting settings

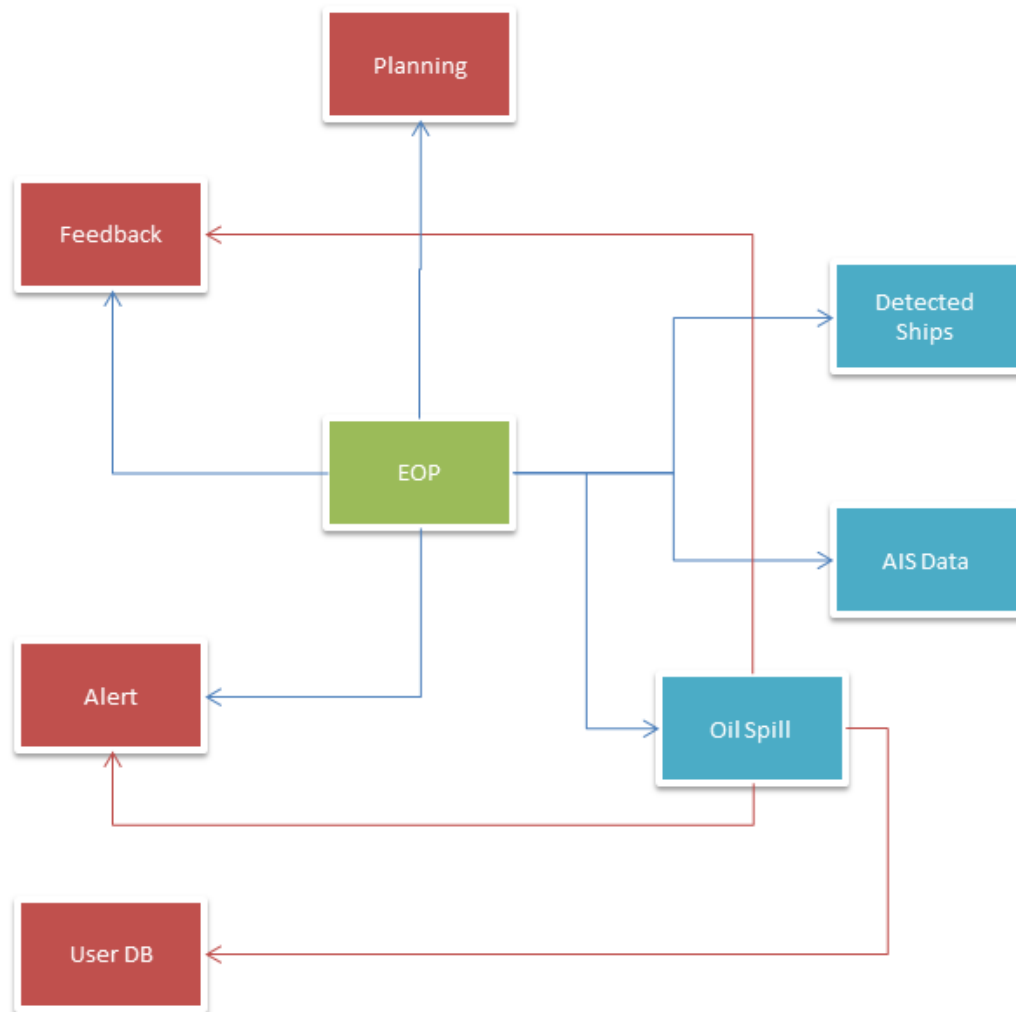


Figure 4-20 high level view of the data model accessed by the WUP

These data objects are grouped in different databases, as defined in detail in § 8. In particular:

- Data in blue colour are served using WFS standard (oil spill, detected vessels and AIS Data)
- Data in green are served using the CSW standard (EOP)
- Data in red are not exposed externally as they are mainly internal data needed for the CSNDC business functions

The links between the various data objects shall be intended as logical links, i.e. links that are exploited when the data are handled inside the CSNDC. The most typical data searches are performed using EOP-centric approach and Oil Spill-centric approach. The logical links are indicated with blue arrows for the EOP and Red arrows for the Oil Spill.

This figure allows to understand the complexity of the relationships among the data. To give an example, when the user searches for EOP data, the data retrieval is not simply a query over the EOP data object alone, but entails the following macro-steps:

- Retrieves EOP data from the catalogue
- Retrieves POR tasked scenes (and merges them with the EOP data catalogue, basically filtering the duplicated)
- Retrieves the corresponding oil spill from the oil spill database
- Retrieves the corresponding alerts from the alert database
- Identifies the AIS data link
- Identifies the Detected ships link
- Identifies the feedback link

Likewise when the user searches for oil spills, the following macro steps are performed:

- Retrieves the oil spill data
- Retrieves the link with the feedback
- Retrieves data from the alert data object
- Retrieve coastal state setting from the user database
- Combines the information from the last 2 bullets to derive the alert level for each oil spill

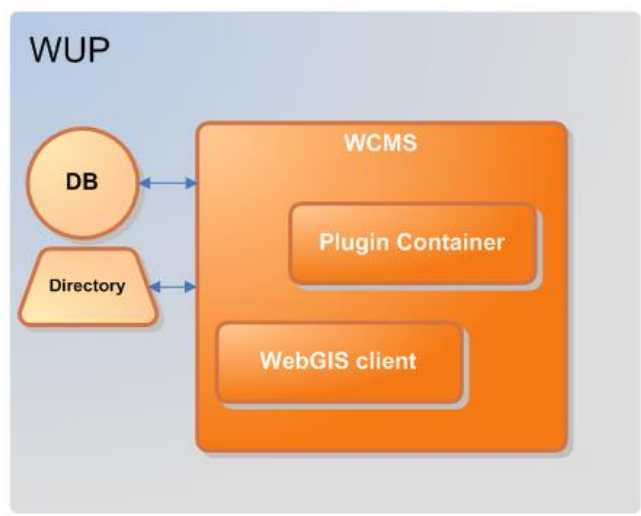
It is clear that the above procedures are not simply individual data requests but a complex combination of data requests and operations over different data object and different databases. This explains why, the data search for the internal business operations of the GIS viewer cannot be easily implemented using the standard OGC services (just to make an example it would not be possible to implement a join between multiple tables).

This approach and the number of links among the various tables also allow to perform cross data navigation, according to the following paradigm:

- EO scenes->
 - Oil Spills
 - Detected Vessels
 - AIS Vessels
 - Feedback(s)
- Oil Spills->
 - EO Scene
 - Feedback(s)
 - AIS Vessels
- Detected Vessels->
 - EO Scene

Looking closer at WUP design, as illustrated in the next figure, it is composed of the following modules:

- Web Content Management System (WCMS)
- Plugin Container
- WebGIS application



4.6.1 WCMS

WCMS is a Web **Content Management System** (i.e. a Web application for creating and managing HTML content and control page flow) and a **Web Portal framework**.

WCMS facilitates content creation, content control, editing, and many essential Web maintenance functions.

WCMS also provides authoring tools to allow users to manage content with relative ease of use in a What-You-See-Is-What-You-Get paradigm. A presentation layer displays the content to users based on a set of configurable templates.

WCMS is based on **Liferay Portal** (www.liferay.com), a Java/J2EE enterprise web portal system that provides personalization, web email, blogs, document library, WCMS, instant messaging, message boards, Wiki and many other web tools available.

The Application Server in which the Liferay Portal solution is deployed is Weblogic.

4.6.2 Plugin Container

Plugin Container is the module that allows for the integration of external portlets, web applications and or user interface software components. Through the Plugin Container COM, JOU, POR, UMA and PDE web applications and portlets are aggregated into the WCMS portal. Liferay Portal integrates a Portlet Container that supports open standards Java Portlet Specification JSR168 and JSR286.

4.6.3 WebGIS application

WebGIS application is the rich internet application that provides advanced user interfaces for geo-spatial data visualization, browsing and access.

It is based on a multi-channel php framework for web GUIs called "SIBILLA".

SIBILLA (Solid Internet Based Interactive Long Lasting Applications) is a unique framework that allows programmers to rapidly develop and deploy robust state-of-the-art web-based applications. It is completely developed by Advanced Computer Systems S.p.A. and is currently used for most of ACS's web-based interfaces.

It is a multi-channel framework in the sense that it has been designed to serve different client technologies or "channels". The framework in itself, unaware of the specific nature of the channels being served, handles GUI content and user interactions as generic ACTIONS that are technology-independent. It is the specific client channel that is in charge to "translate" the meaning of the ACTIONS to its specific technological context and to adapt them to its own operational environment. Currently, two web channels are supported: a rich flex-based client and a plain javascript-based thin client.

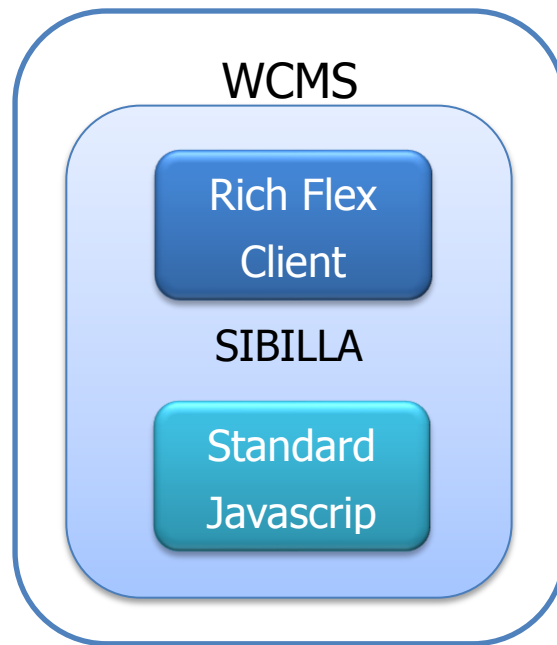


Figure 4-22 SIBILLA based channels

The main channel is supposed to be the rich flex client that leverages Adobe Flash plug-in, the standard javascript client being reserved to such rare cases in which the end-user environment does not allow for Flash plug-in installation.

The rich flex client is made of a multipurpose flash movie called sibilla.swf. That is the SIBILLA dedicated hub that interacts with SIBILLA framework for “building” at runtime a context-specific web GUI as depicted in following sequence diagram:

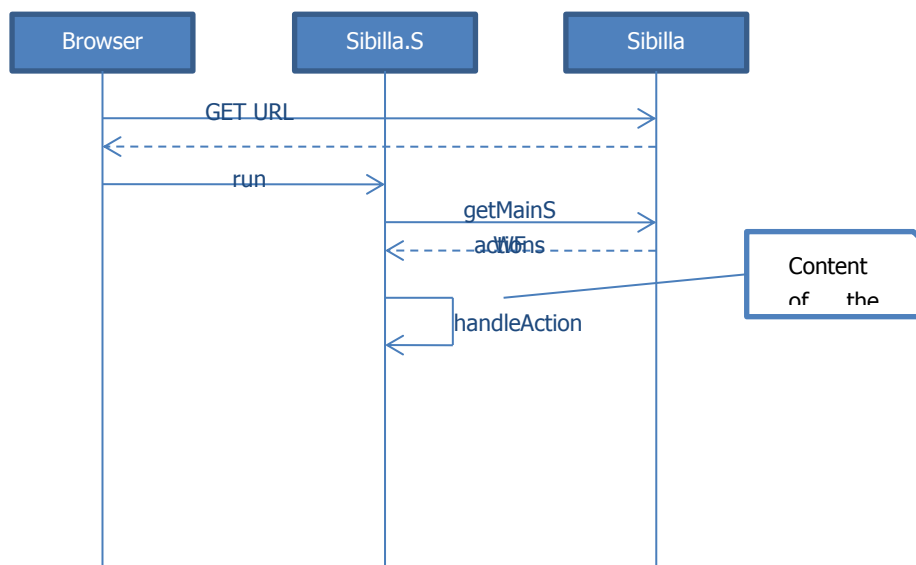


Figure 4-23 Rich flex client init sequence

When the browser activate the sibilla.swf flash movie, it asks SIBILLA for its context-dependant content (getMainSWF() call). The SIBILLA layer, then, responds with a number of ACTIONS (channel independent) that instrument the GUI content and behaviours to be created. At this point the Sibilla.swf movie interprets the given ACTIONS and build its own version (channel specific) of the graphical content.

In doing so, Sibilla.swf makes use of a number of advanced Flex components among which:

- ORUS, a map viewer that acts as a WMS/WFS/CSW configurable client
- DataGrid, an advanced grid component for tabular data representation

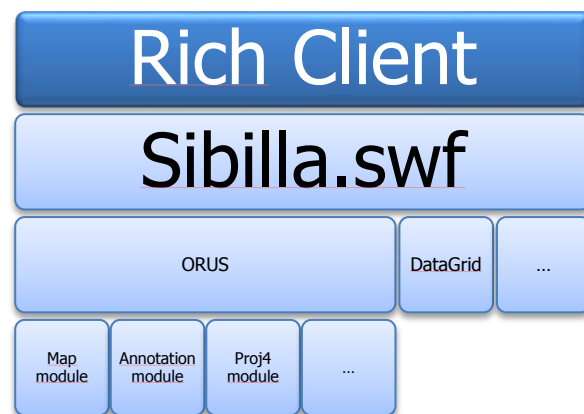


Figure 4-24 RichFlex client decomposition

Each component can make use of specific modules as needed in a plugin approach. For example, the ORUS component can make use of a Map Navigation Module, a Map Annotation Module, a geographic Projection Module and so on. Each module can be activated/deactivated/configured independently.

The standard javascript channel shares the same high-level structure.

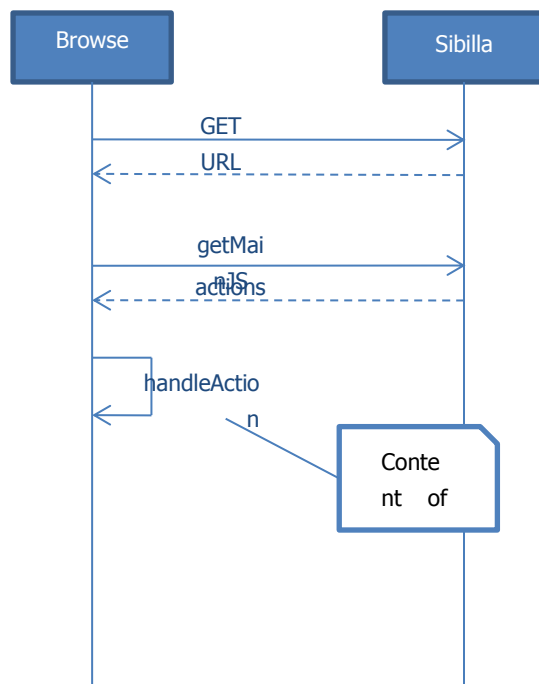


Figure 4-25 Standard javascript client init sequence

When the browser starts a plain javascript channel, it downloads an html javascript-enabled page with a main function that asks to SIBILLA for its context-driven content (getMainJS() call). The SIBILLA layer, then, responds with ACTIONS (channel independent) mapped to channel-specific javascript components and functions to instrument the GUI content and behaviours to be created.

The standard client is based on Mapfish an open source javascript toolbox for creation of complex html/ajax/javascript components for map viewing and tabular data presentation. It is based on a number of lower level libraries of Javascript layers such as ExtJS, OpenLayers and GeoExt.



Figure 4-26 Library hierarchy for the standard javascript client

Please refer to Functional Design Document section 9 “Wireframe Definition” for a description of main GUIs presented in the WUP component leveraging SIBILLA.

4.6.4 Oil spill prediction models integration from the WUP

This section presents an overview of the integration of the manual workflow for triggering and displaying the results of Oil Spill propagation /and back-propagation models), hereinafter simply referred to as *oil spill models*.

The high level workflow for running oil spill models is summarised hereafter:

- User searches for oil spills and selects the details oil spill of interest
- From this view, the user trigger an action to run the oil spill models
- The user selects the model to run, out of a list of candidate oil spill models, fill in the necessary input parameters and triggers the very model execution
- The user monitors the status of the model processing and of the ingestion of the modelling results
- Upon successful processing and ingestion of the processing results, the user displays the results of the model simulations in the GIS viewer in the form of an animation and/or selecting individual model simulation steps (the oil spill models simulate the propagation in time of a given oil spill, with a number of predefined time steps, e.g. every 15 minutes)

Details of this implementation are reported in the sections 5.10 below.

4.7 PDE – Product Delivery

4.7.1 PDE Overview

The Product Request and Delivery component, also referred as PDE, has the following responsibilities:

- Allow subscriptions for services data sets
- Generation of reports and alerts

4.7.2 PDE Decomposition

The PDE decomposition is reported in the following figure. The PDE is devoted to the output communication channels towards the external users of the CSN-DC, typically the coastal state users. There are basically two types of outputs:

- Delivery of products
- Delivery of reports and alerts

The 2 types of output are significantly different in terms of scenario. In the former case, the users identify some products of interest and request access to products. In the latter case the user is routinely informed with reports and alerts about the situation of the CSN-DC analyses. The service subscription manager and the data dissemination proxy are basically devoted to the user data access.

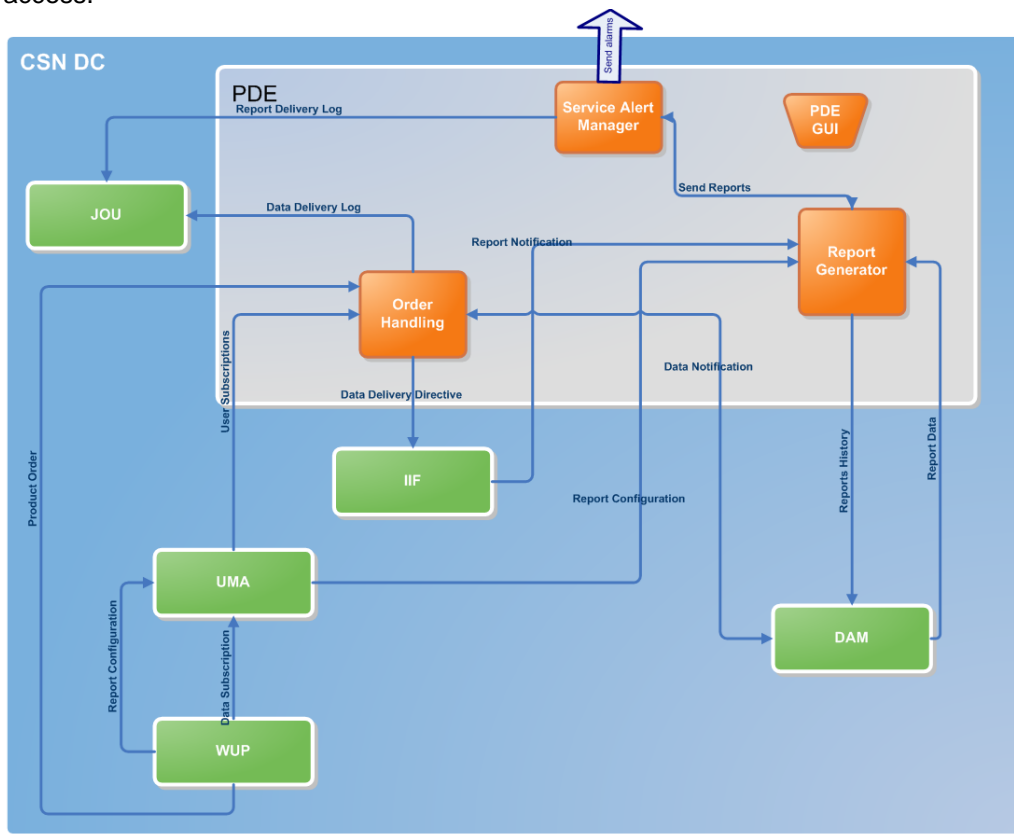


Figure 4-27 – PDE Decomposition

Order Handling

This service will orchestrate the delivery of products to the user on the basis of existing subscriptions. Each time there are products to which at least one user is subscribed, the DAM, after successful ingestion of the products, notifies the data reception to the Order Handling in order to initiate the data delivery procedure. The Order handling retrieves the user details from the UMA component and instructs the IIF for final dissemination of the products to the users who subscribed.

There are basically 2 types of subscription:

- Subscription for EO products (which will be delivered via GeoTIFF format)
- Subscription for oil spill data (which will be delivered in shapfile format)

Regarding the distribution protocol there are 2 possibilities:

- Email distribution
- FTP distribution

The user will have to set following parameters:

- Product type (oil spill or EOP)
- Subscription criteria (it can be a combination of time window and a geographic area)
- Distribution format (email or FTP)

When the user chooses distribution via FTP, an email account shall always be provided in order to notify the user of the successful distribution. When email distribution is chosen, if the file type is oil spill, this is distributed as actually an attachment to the email to the user, while in case the user chose the distribution of EOP, the email will simply contain a link for downloading the product.

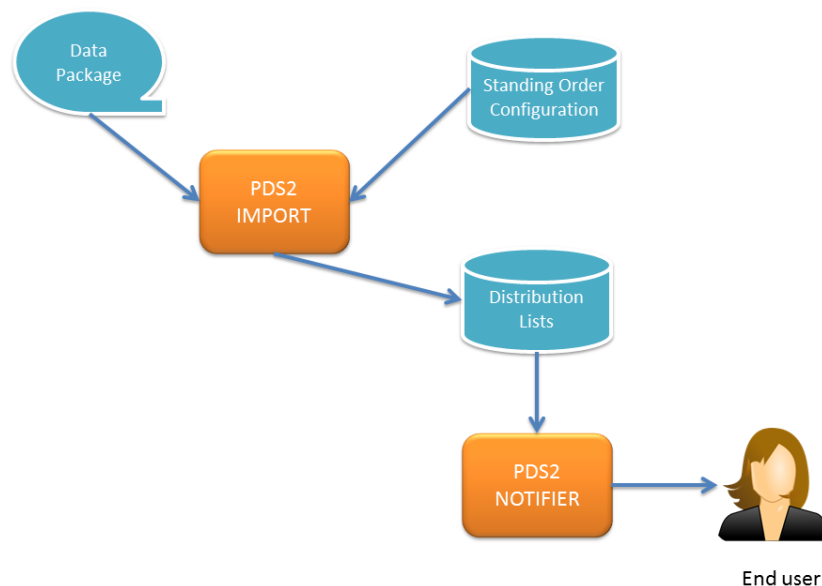


Figure 4-28 Standing order flow

The standing order implementation is described in Figure 4-28. The process is rather simple and divided in 2 main asynchronous steps:

- PDS2² Import
- PDS2 Notifier

It is assumed that users have made their configuration of standing orders, by configuring all the parameters described above. This will be stored in the standing order configuration table.

The *PDS2 Import* will do the following:

- For each incoming product:
 - Check if the product matches the standing order criteria (type geo and time) defined by at least one user (it can match one or multiple users)
 - If the product matches at least 1 set of criteria, the product is eligible for notification therefore:
 - the info is registered into the Distribution Lists
 - the product to be distributed is created in the appropriate format (e.g. GeoTIFF for EO data or Shapefile for OSN)

The *PDS2 notifier* will do the following:

- Routinely (activated by a cron job) checks the to-do-list in the distribution lists table
- If there are relevant events, implements them

A typical event to be implemented by the notifier is:

- Send the needed data by FTP + notification
- Send the data via email (in attach or with a link for downloading the data in case of EOP)

Report Generator

Reports are a fundamental output of the CSN-DC system as they are the ultimate result of the analysis of the data generated from the SPs possibly integrated with further systematic and/or ad-hoc analyses performed within the CSN-DC (integration with vessel traffic/Meteo/Oceanographic data, execution of ad-hoc oil spill models, etc.). The content and format of the report may vary depending on the following variables:

- Type of scenario (oil spill detected, clean sea, vessel detected)
- Characteristics of the oil spill (confidence level, distance from the coast, size, etc.)
- User configuration (preferred format and type of report)

The Report Generator is systematically triggered by the IIF on the following cases:

- Upon reception of an OSW an alert of type oil spill warning is sent
- Upon reception of an OSN an alert of one the following types will be sent:
 - Oil spill notification if there is at least 1 oil spill in the area of the end user
 - Clean sea report if there is no oil spill in the area of the end user

The alerts are managed taking into account the following concepts:

- Alerts are sent to end users
- Each end user has one or many areas, defined as *Alerting Areas*

² The name PDS2 derives from historical reasons, as this component was delivered for the Second Generation of PDS in ACS.

- Alerts are sent to the end user as long as the footprint of the received EOP data intersects at least one of the user alerting areas
- If the footprint of the image intersects the Alerting Area, but the oil spill present in the image is outside the Alerting Area, the user will receive a notification of type *Clean Sea* (indicating that no oil spill is present in the area of interest)
- Reception of a data package including oil spills (warning and notification) can lead to distribution of alerts to multiple users (this is the most frequent scenario, since image footprints are typically intersecting multiple user alerting areas)
- The alert report contains a lot of detailed information for each oil spill plus additional information (e.g. meteo, vessel data, etc.)
- Each oil spill contained into an alert report is classified into 3 levels of severity (red, yellow, green), which are computed by a combination of parameters extracted from the oil spill incoming package and computed on the fly by the system
- The end users have the capability of setting the weights for each of the individual parameters that are used for computing the severity of a given oil spill
- As a result of the previous point, the severity of each oil spill for different users can be different
- In general terms, the reception of oil spill warning and notification package leads to creation of multiple alert report, whose content (e.g. severity level, number of oil spills intersecting the area of interest, etc.) are very different from each other

Moreover in terms of distribution characteristics, each user will have the capability of:

- Defining the list of who within the user's coastal state will receive the notification
- Defining the severity level upon which notification will be sent
- Defining the distribution mechanism (e.g. email, sms, phone call³)

All of the above implies that:

- There must be some complex computation which is performed when each package is received which:
 - Extracts the relevant information from the oil spill input data
 - Computes the necessary additional info (e.g. distance from coast, etc.)
 - Identifies the areas intersecting the footprint
 - Calculates the characteristics of the alert for each coastal state (i.e. end user organisation) whose alerting area intersect the footprint
 - Stores some intermediate data for report generation

The Report generation process is summarised in the following figure.

³ Even if originally foreseen, sms, mms and phone call delivery mechanisms have not been implemented by any automated mechanism. They are indeed implemented by a human operator in EMSA who will receive an email with the characteristics of the alert and will implement the actual notification (e.g. by calling the end user).

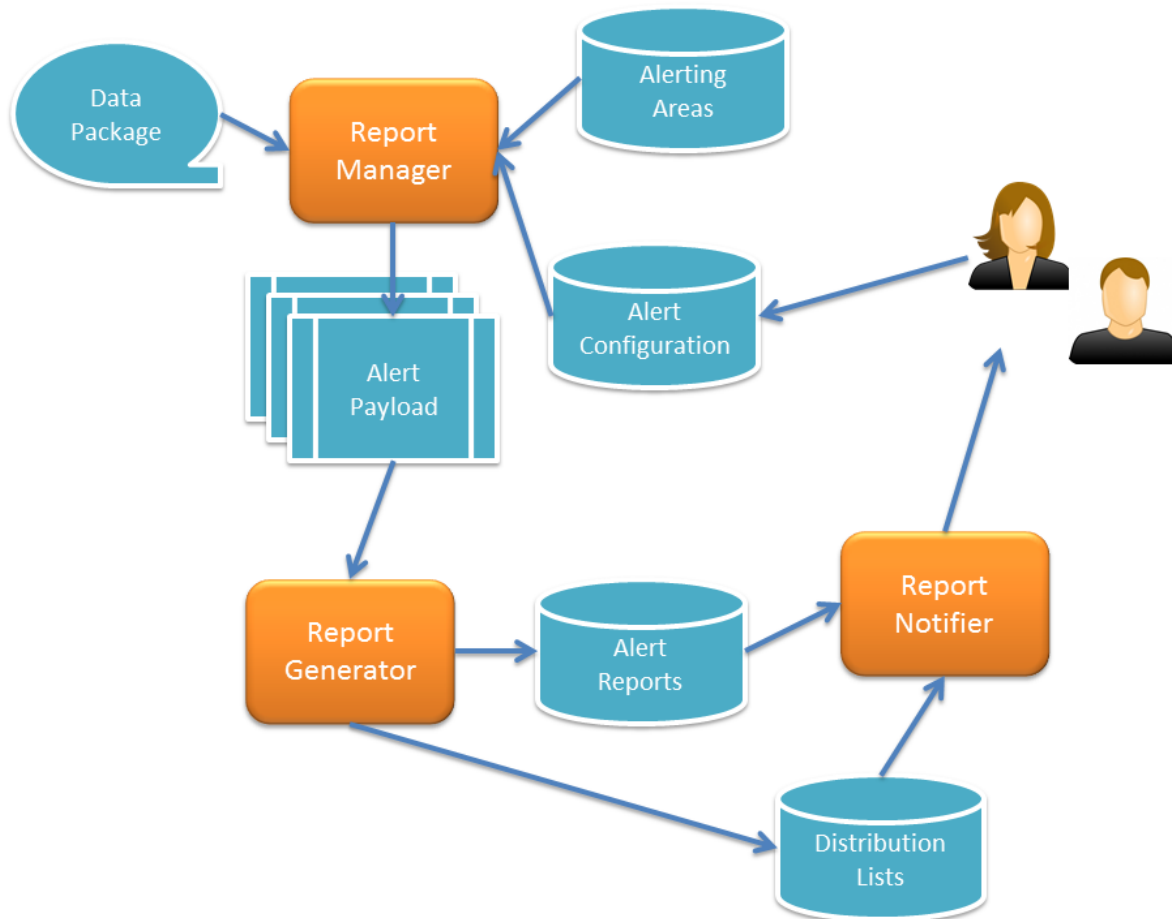


Figure 4-29 - Report generation workflow

The Report Manager is triggered by the arrival of a data package of type OSN or OSW. On the basis of the configuration of the alerts (previously defined by the users, or by the EMSA administrator on behalf of the users) and the alert areas for each end users, the Report Manager computes the specific information that is needed for each report. There will be 1 report generated per received file and per interested coastal state (e.g. footprint intersecting the area). The result of this analysis is the *Report Payload* (physically this is implemented by an XML file which contains all relevant information for creating a report for each relevant coastal state). For each payload generated the Report Generator is invoked which performs the following operations:

- Creates the actual PDF report using the Jasper Report COTS and stores it into a local table (*Alert Reports*)
- Creates a corresponding entry in the distribution lists table

Finally the *Report Notifier* will routinely check the distribution lists and, if applicable, send PDF reports to the end users according to their respective configurations.

For more details about this process please refer to:

- Annex B (§ 12) for a list of parameters to be extracted and computed by the Report Manager

- Reference doc [ALR] for a details composition of the PDS report

4.7.3 Details on the alerting procedure

Since the alerting is a fundamental component of the CSNDC, a specific chapter is included here in the Logical view where it is described in terms of component and interaction diagram.

Here follows the Alerting component diagram.

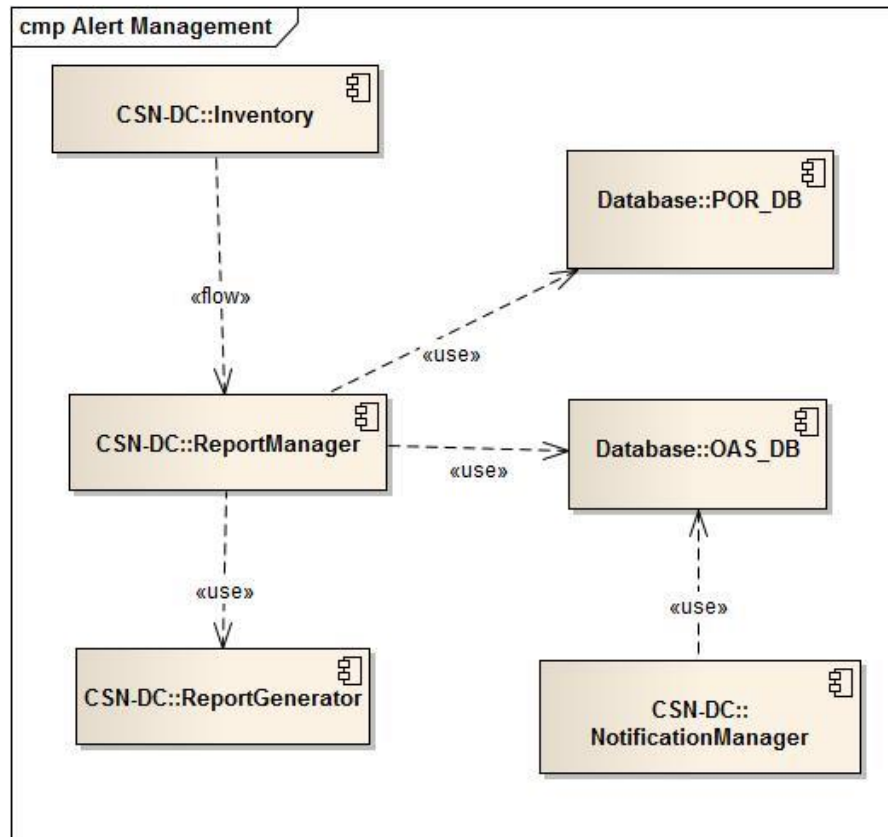


Figure 4-30 - Alert Management Component Diagram

Management of the alerts entails a complex exchange which includes the following components:

- *Inventory*: ingests the OSN data packages and, upon reception of oil spill data triggers the ReportManager
- *ReportManager*: is the core of the reporting system. It exploits the information in the OSN plus the information stored in the various databases, in order to prepare all the data necessary for creating an alert. It is the business component of the reporting
- *ReportGenerator*: this component uses the JasperReport COTS to generate the actual PDF report which will be disseminated to the end users
- *POR_DB*: it is the POR database, and stores information regarding the scenes, such as the footprint geometry and other metadata.
- *OAS_DB*: it is used for supporting all business functions of the alerting, including:
 - stores the configuration of the alerting settings of the various coastal states

- stores the info about the reports to be disseminated and the result of the dissemination
- *NotificationManager*: uses the information stored in the OAS_DB to distribute the emails to the coastal states with the corresponding PDF alerts reports in attachment

The Alerting workflow can be better clarified by analysing the following dynamic model.

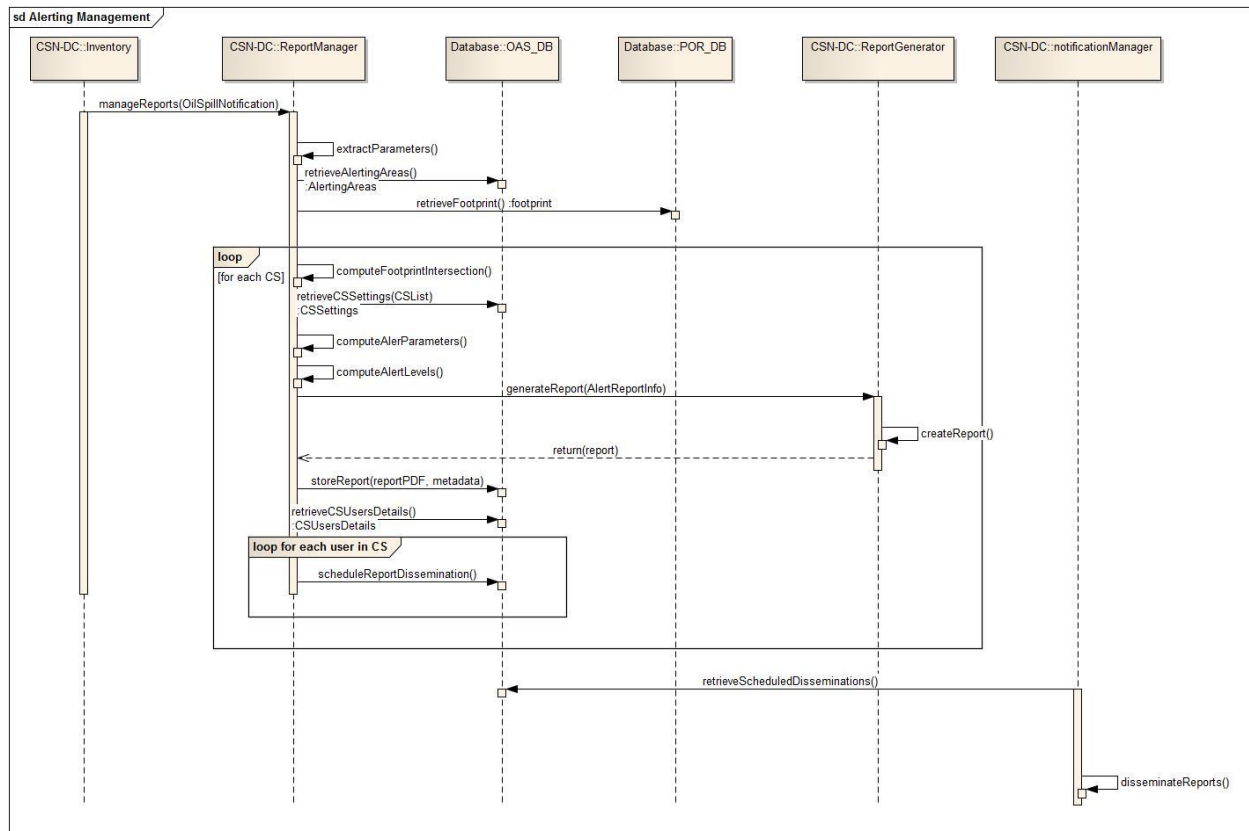


Figure 4-31 - Dynamic model of the alerting process

The following actions are performed:

- The Inventory, upon reception of a package of type OSN or OSW, triggers the ReportManager, to which it sends the relevant package information
- The ReportManager performs the following operations
 - Extracts the relevant parameters from the data package (e.g. oil spill geometry, data sources, clip images, meteo data, etc.)
 - Retrieves the alerting areas from the OAS_DB
 - Retrieves the footprint from the POR_DB
 - Perform the intersection between the footprint geometry and the alerting areas
 - For each coastal state whose alerting area is intersecting the image footprint:
 - Retrieves the coastal state settings (e.g. alert severity levels, thresholds, print preferences, etc.)
 - Computes a number of alerting parameters that are not directly extracted from the oil spill notification/warning data (e.g. distance from the coast line, distance from sensitive areas, etc.)

- Computes the alert level for each oil spill, by integrating the coastal state setting with the parameters extracted from the data and computed in the previous step
 - Prepares the relevant data and invokes the ReportGenerator to create the PDF report by combining.
 - The ReportGenerator exploits JasperReport COTS and combines the data received from the ReportManager with a predefined set of templates to create the PDF report
 - The ReportManager receives the PDF created by the ReportGenerator and stores it in the OAS_DB
 - The ReportManager loop for all user configured in the communication matrix for each coastal state to store the distribution records in the OAS_DB
- Upon regular time intervals the NotificationManager retrieves the list of reports to be distributed from the OAS_DB and performs the actual distribution according to the distribution channel (e.g. email)

The decoupling of the NotificationManager from the creation of the reports is a typical pattern which is recommended when information shall be distributed via some channel (e.g. e-mail, file, etc.). This mechanism allows to create the distribution queue and, even if the distribution channel is not available (e.g. could depend on availability of the mail server, etc.) the information is not lost, but it is kept in the queue, meaning that it will be successfully dispatched when the unavailable service will become available again. Similarly, this also allows to implement retry mechanisms.

4.8 POR – Planning and Ordering

4.8.1 POR Overview

The planning and ordering component (POR) is the sub-system which is **responsible for managing the workflow for acquisition of EO data**. The procedure for ordering SAR data is rather complex and involves a number of different actors, in a workflow which is finally aimed at optimising the procedure both from the budget and from the final efficiency point of view. The overall planning of image acquisition involves the Coastal State (CS) users and is called image allocation.

The procedure is driven by the coverage requirements defined by the CS, which are defined in terms of areas to be covered, coverage frequency, coverage density, etc. These requirements are translated into actual SAR images to be acquired by the CSN Service Desk (CSD), who are the CSN operators responsible for carrying out this task. Using various mission planning tools, these operators shall define the exact SAR scenes, corresponding to future acquisitions, which are necessary for covering the monitoring requests of the CS. These scenes represent potential acquisitions, because they need to be confirmed by the **Satellite Operators (SO)**, who shall verify that the requested scenes are not in conflict with other planning requests having higher priority. Moreover the feasibility of SAR data acquisition shall also be confirmed by the **Service Providers (SP)**, who shall verify the compatibility of the requested scene with the foreseen ground station programming for scene acquisition. The selected list of SAR scenes shall be made visible to the CS, who will make the final selection of the scenes that they allocate for the service provision. The decision of the CS will obviously be made also on the basis of budget considerations, involving both the cost of the data and the cost of the service (which in principle is correlated with the number of processed images).

It is important to describe a few key concepts of the POR:

- The workflow for planning and ordering data acquisition is managed using the concept of a *scenes* and *order*. An order is similar to a shopping cart in a typical e-commerce system (in the POR system the terms cart and order are sometimes interchanged), therefore it is a collection of ordered items. The scene is an individual ordered item. Typically the POR is managed by orders that includes scenes that are logically grouped (e.g. grouped by months which are planned in advance, or simply grouped by other criteria, i.e. they are ordered for the same purpose, etc.). Some aspects of the POR workflow refer collectively to the order (e.g. generation of tasking forms, while others are referred to the single scene, e.g. allocation)
- The workflow implemented by the POR is managed by a number of actors, as indicated above. Each actor has a clear role in the process and have different visibility and functional rights.
- The POR is tightly interlinked with the JOU and FINSYS systems, therefore a number of messages are exchanged between the POR and the JOU/FINSYS during the whole POR workflow. Interactions with the JOU are mainly related to the communication of information about the ordered scenes (that must be later used for comparing the expected data with the data actually received, including timeliness, etc.) . Interactions with the FINSYS are rather oriented towards computation of budget information and creation of the tasking forms that shall be used for the formal ordering process.

4.8.2 POR Main tasking phases

The POR planning and ordering is also referred to as *tasking* using a collective terms which refers to the whole end-to-end process.

The main tasking phases are:

- Planning
- Tasking allocation
- Approval

Planning and tasking allocation are managed on individual scenes, while the approval phase is managed by an order as a whole. Some more details of the various phases are reported in the following subsections.

4.8.2.1 POR Planning phase

The planning phase is rather simple, is only managed by the EMSA Service Desk and consists in the following operations:

- Upload of a planning file
- Selection of the scenes that are candidate to be tasked
- Starting the tasking phase

The upload of the planning file is performed by loading on the POR a file in one of the following formats:

- EOLI
- ACP
- SwathPlanner
- Savoir (.csv)

Please refer to 13 for some examples of the files managed by the POR.

Once uploaded on the POR, using the planning file, the POR loads all data details into its internal database and uses this information for managing the rest of the workflow, and to compose the messages that are distributed to the JOU/FINSYS.

The EMSA SD user may select one or multiple scenes from the POR and start the tasking. When this occurs the POR creates the list between the *order master* (the information collectively related to a given order, such as status of the order, label, etc.) and *order details* (the one-to-many link to the details of each individual scene of a given order). Order details include the details of each individual scene, as inherited from the planning file, plus the various info regarding the scene lifecycle, e.g. the current status.

4.8.2.2 POR Tasking allocation phase

This is the most complex phase from the point of view of the workflow and starts once the EMSA user has closed the planning phase, as described above. The various stages are schematically illustrated in the following figure.

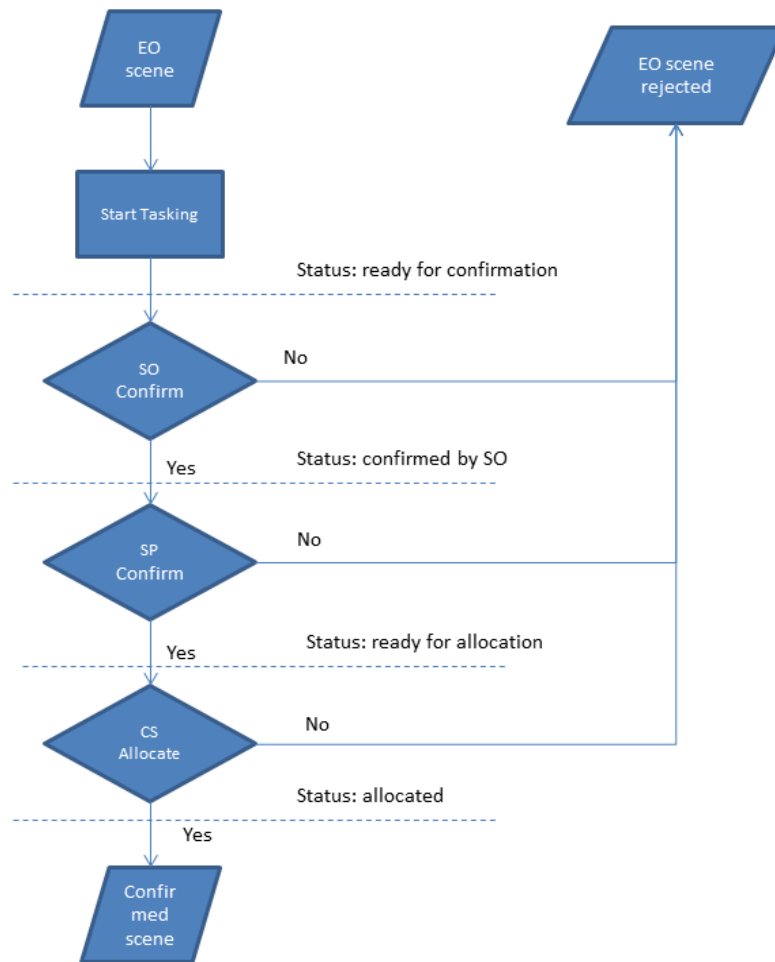


Figure 4-32 - POR Tasking phase

A given EO scene starts the tasking phase, after the EMSA SD user has selected that scene in the planning phase, before triggering the *Start Tasking* operation. This sets the scene in status *Ready for allocation*.

At this stage the scene is visible to the users of type *Satellite Operator* (aka *License Provider*) and *Service Provider*. Both user types can change the status on their own at the same time (there is no order of precedence). Once the scene has been confirmed by both SO and SP, it goes into the status *Ready for allocation*. In this status it can be allocated by the Coastal States. After has been allocated by the coastal states, it goes into the status *Allocated*. If a user does not confirm/allocate the scene, it goes into the status *rejected*, where it cannot be used for the tasking. A few more details on this workflow:

- While the actions of confirmation by means of SO and SP are mandatory before the scene becomes available to the CS for allocation, there is no order or precedence between SO and SP. Both can access the POR and confirm the scene on their behalf at any stage.
- The confirmation of SO and SP can be reverted forth and back. This is true until the CS has allocated the scene (in which stage the SO/SP can no longer dis-confirm the scene).
- Also the CS allocation can be reverted back and forth, but only until the final approval phase has started.

Once a number (1 or many) of scenes have been finally confirmed by all actors, the EMSA SD user can start the approval phase, described in the next section.

4.8.3 POR Approval phase

The approval phase is depicted in the following flow chart.

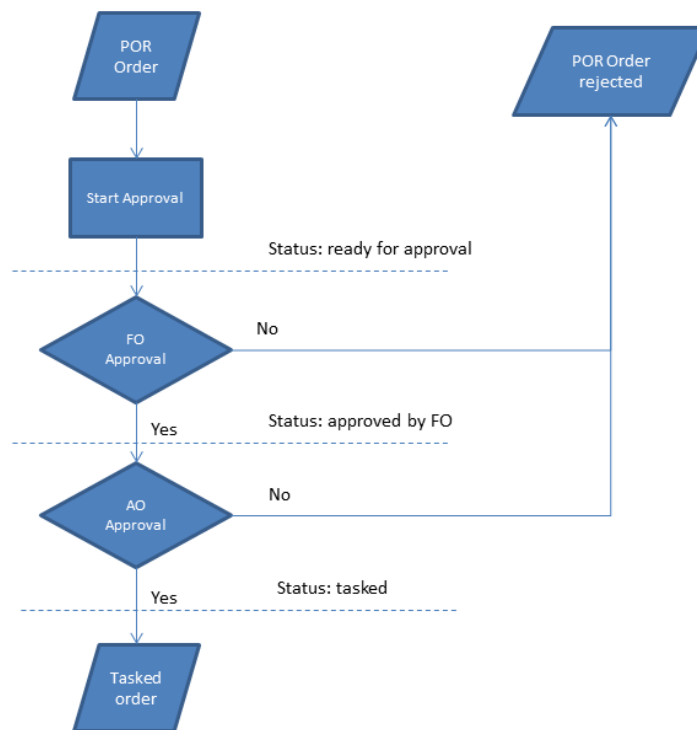


Figure 4-33 Flow chart illustrating the approval phase

As already indicated above, the approval is carried out for a given order in bulk. The order undergoing approval is composed of the scenes selected by the EMSA SD user, out of those that are in status *allocated*. When the start approval process starts the order goes into the status *Ready for approval*. At this stage the Financial Officer (FO) has the following options:

- View financial info
- Approve tasking
- Reject tasking

The first option is always mandatory (not explicitly mentioned in the figure above, because it does not change the status of the order, but simply it is a mandatory step for the user to be allowed to execute any of the other 2 actions, i.e. approve/reject. If the order is rejected, it goes back to the previous stage and could potentially be resubmitted by the EMSA SD for approval (supposedly after applying some changes, e.g. reducing the number of scenes, etc.). If it is approved, it goes into the status *Approved by FO* and is ready to be approved by the Authorising Officer (AO).

A similar logic applies to the following macro-step. The authorising officer is required to show the financial info before being allowed to either approve or reject a given order. After having viewed the financial info, if rejecting the order, it goes back into the previous status, while if approved, it triggers the order finalisation, which consists in:

- Creating order and tasking forms for the various stakeholders (e.g. SO, SP)
- Sending them to the appropriate recipients

4.8.4 Usage of Service Types

The CSNDC allows to implement a flexible definition for service type.

A **service type** is defined as a possible combination of **service elements**. Service elements are individual service tasks that shall be performed. There are ALWAYS 2 service providers involved in the provision of a service to EMSA. They carry out their task in sequence, so that there will always be Provider-1 and Provider-2, whereby the former performs its tasks on a given service ID (i.e. a given scene) prior to the tasks to be performed by Provider-2. This could be considered as a processing chain where a first set of steps is performed by Provider-1 and a second set of steps are performed by Provider-2. Provider-1 and Provider-2 will exchange data by independent means, but the time of exchanging the data between 1st and 2nd provider shall be traced in order to be able to trace the QoS (timeliness in this case) for both.

The possible list of service elements to be carried out by **Provider-1** is reported below:

1. Licence
2. Data downlink
3. Image processing
4. Image delivery

The possible list of service elements to be carried out by **Provider-2** is reported below:

1. Data downlink
2. Image processing
3. Image delivery
4. Oil spill detection
5. Vessel detection
6. Activity detection

Moreover the user will have the capability of setting (optionally) a **Maximum delay time** (DMAX, this also applies by default to all services with a given cart, and can be overridden for each individual scene). There will be a delivery time for both provider 1 and provider 2.

The **services type** is made by a combination of **2 service elements lists**, one for each provider type (i.e. provider 1 and provider 2), **the names of the provider assigned to each list** and other info such as delivery delays, optional elements, etc. The name of provider-1 and provider-2 can actually be the same. The POR user shall be able to create service types using a UI which allows to:

- 1) Define service name
- 2) Define service description
- 3) Define Max Downlink Delay (in minutes)

- 4) For Provider-1
 - a. Choose the provider name (or set automatic mode, which will utilise a predefined *look up table*, that maps platform name into the Provider-1, where platform name is unique key).
 - b. choose the list of service elements from the available service list
 - c. Optionally choose one of the Special Fee attributes, out of the following list:
 - i. Archive Image
 - ii. Emergency Programming Service
 - iii. On Board Recorder Usage
 - d. Choose the *Max Delivery Delay*
 - e. Choose Short Notification Threshold (in decimal hours)
- 5) For Provider-2
 - a. Choose the provider name (or set automatic mode, which will utilise the Tasking Area algorithm already in place as of Release 1.6).
 - b. choose the list of service elements from the available service list
 - c. Optionally choose one of the Special Fee attributes (initially his list is empty, but it could be expanded in the future)
 - d. Choose the *Max Delivery Delay*

The service type chosen when starting a tasking will apply automatically to all scenes in the cart. The POR user may choose one or multiple scenes in the cart and override the service type assignment (choosing from the service types available).

The following figure shows a preview of the UI to create and edit a service type. Please note that the Provider name field is mandatory, but in fact there is drop down with a number of options, one of which is *Automatic* and the others are the list of available Providers.

Define Service Type

Service Name
Standard

Service Type Description
test

Max Downlink Delay (minutes)
0

Notification Threshold (decimal hours)
0

Provider-1 Service Definition

Provider Name
Automatic

☐
☐

service elements

☒
License

☒
Data Downlink

☐
Image Processing

☐
Image Delivery

☐
☐

additional service elements

☒
Emergency Programming Service

☒
On Board Recorder Usage

☐
Archive Image

Max Delivery Delay (decimal hou...
0

Provider-2 Service Definition

Provider Name
Automatic

☐
☐

service elements

☐
Data Downlink

☐
Image Processing

☒
Image Delivery

☐
Oil Spill Detection

☐
Vessel Detection

☐
Activity Detection

Max Delivery Delay (decimal hou...
0

Save
Close







Figure 4-34 – UI for defining service types

The POR user with appropriate privileges (i.e. UP02 or UP21) will be able to access a list or pre-existing service types, which will be displayed in a table. Initially the POR will be installed with a number of pre-defined service types to be provided by EMSA. The privileged user will be able to select a given item in that table and view and possibly edit the details, using a UI similar to the one shown above. Additionally the user will have the capability of clicking on an *Add* button, which will open the same UI again, but with all empty fields. Using this UI the user can create a new service type and add it to the list. Likewise the user may select one or multiple service types and delete them explicitly.

When starting a tasking, the user will be prompted with a dialog like the following.

Tasking

Service Types

	Service Name	Service Type Description	License Provider	Service Provider	Created by	Created on
	EGEOS service		Automatic	EGEOS	XE-UP21-01	2014-07-11 14:46...
	SO - SP defined		ESA	EGEOS	XE-UP21-01	2014-07-11 08:58...
	SO Automatic		Automatic	CLS	DEL_SD1	2014-07-11 08:53...
	SP Automatic	servizio da tenere per non av...	ESA	Automatic	DEL_SD1	2014-07-11 08:51...
	Andy	For All	Automatic	Automatic	XE-UP21-01	2014-06-16 12:10...
	Standard	test	Automatic	Automatic	DEL_SD1	2014-06-11 10:31...

Order Label

SP-1 Delivery Timeout (days)

SP-2 Delivery Timeout (days)

CS Delivery Timeout (days)

Operation

Mode

Figure 4-35 – Dialog window for defining service type for a given cart

The user will be able to choose the service type to associate to a given cart. All scenes of that cart will be set to that service type.

Moreover, once the tasking has started, the user may choose to change the service type for that particular set of scenes in the *tasking* tab (see §4.8.2.2). This operation will only be allowed if no SP or SO has confirmed the scene. After that it will be no longer possible to make this change.

4.8.5 POR decomposition

The POR decomposition is reported in the following picture.

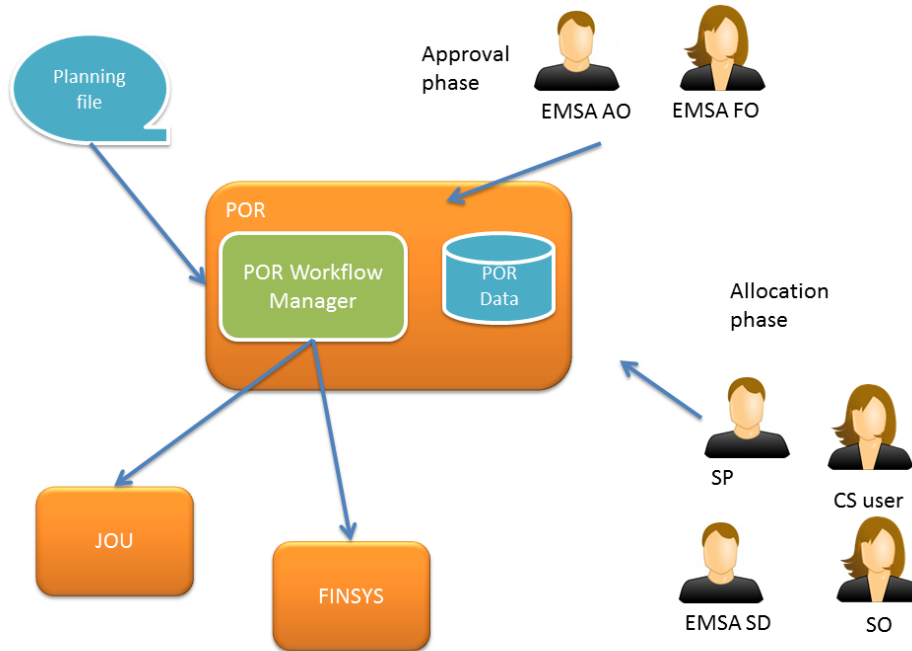


Figure 4-36 - POR first level of decomposition

The **POR Workflow Manager** is responsible for handling all interactions regarding the image allocation. It is based on a state machine which can handle state transitions (requested, planned, rejected, etc.) during the image allocation procedures. The state transitions are governed by transition rules which depend on the type of actor which is logged on the system. State transitions apply to individual scenes of the planning exercise. Each scene can have a certain number of states and the transitions that are allowed depend on the state in which the scene is and on the actor which is logged on the system. This simple and robust mechanism allows to easily manage the interactions between all actors during the workflow. Each state transition shall be logged and stored, so that the state transition history can be traced at any stage. Moreover the **POR Workflow Manager** is responsible for handling a series of import/export functions, which allow to enter the scene list as generated by the output of a certain mission planning tool (e.g. EOLI-SA). These functions allow for easily managing lists of scenes, so that the various actors (e.g. the CSN-SD, the SO, etc.) can use their tools for changing the state of a list of individual scenes. When importing the scene list, with associated a state change, the system shall automatically check for each individual transition whether the scene state change is allowed. This means that the Planning Server shall be based on a stateful implementation which stores the previous state of each scene list. Finally the **Workflow Manager** will have the capacity of allocating scenes to coastal states and to service provider based on geographic coverage mechanisms, to be defined with EMSA. It is assumed that the allocation to coastal states for each image will be based on a vector layer which defines the territorial waters belonging to each country (allowing the possibility of having multiple allocation for scenes spanning across boundaries). Same applies for service analysis allocation to service providers, which will be based on vector files defining the boundary areas of each SP.

The **Workflow Manager** is a fundamental tool for the overall workflow and shall be highly configurable in terms of access privileges, and therefore availability of functions, in order to clearly define the roles of the CSD, SO, SP and CS users. Moreover a notification service shall be put in

place in order to notify the various actors of the status change in the workflow, by configurable means (e.g. e-mail, SMS).

This concept will be further specified in dynamic analysis of the POR reported hereafter (§ 5.7).

The **Planning Web Client**, will be hosted on the WUP. It is an application which allows the users to display the scenes footprints, along with their metadata (e.g. sensor, acquisition mode, orbit, etc.) including the scene status. Moreover the geographical display of the WUP will allow to display the scenes footprints and to overlay the geographical layers which allow to define the scene distribution to the CSs and SPs. Through the **Planning Web Client**, the user can change the states of the individual scenes, or import/export the scene lists into exchange format files.

Once the scenes to be ordered have been consolidated by all parties, the POR will **issue two types of order forms** (these are actually generated and disseminate to the appropriate recipients via the FINSYS as clarified later):

- A scene acquisitions service order to the SOs
- A scene analysis service order to the SPs

The Order Handler manages all the history of the orders and keeps track of this through the **Order DB**, which stored all data about the orders. It will represent the interface with the JOU component for retrieving order history information.

The POR interfaces with 2 internal components:

- With the FINSYS to which provides details of the various scenes ordered with the object to:
 - Create and send tasking forms via email (to SPs and SOs)
 - Store information that will be used for calculating the pricing for the various scenes (on the basis of the configuration of the FINSYS, see the specific FINSYS chapter for more details)
- With the JOU to which the POR sends the log of the order forms generated at the end of the allocation process (see § Figure 5-29)

4.9 JOURNALING

The overall functional component diagram, with the grouping of components into layers for the JOURNALING is depicted in the figure below. The sections that follow the mentioned figure describe those components.

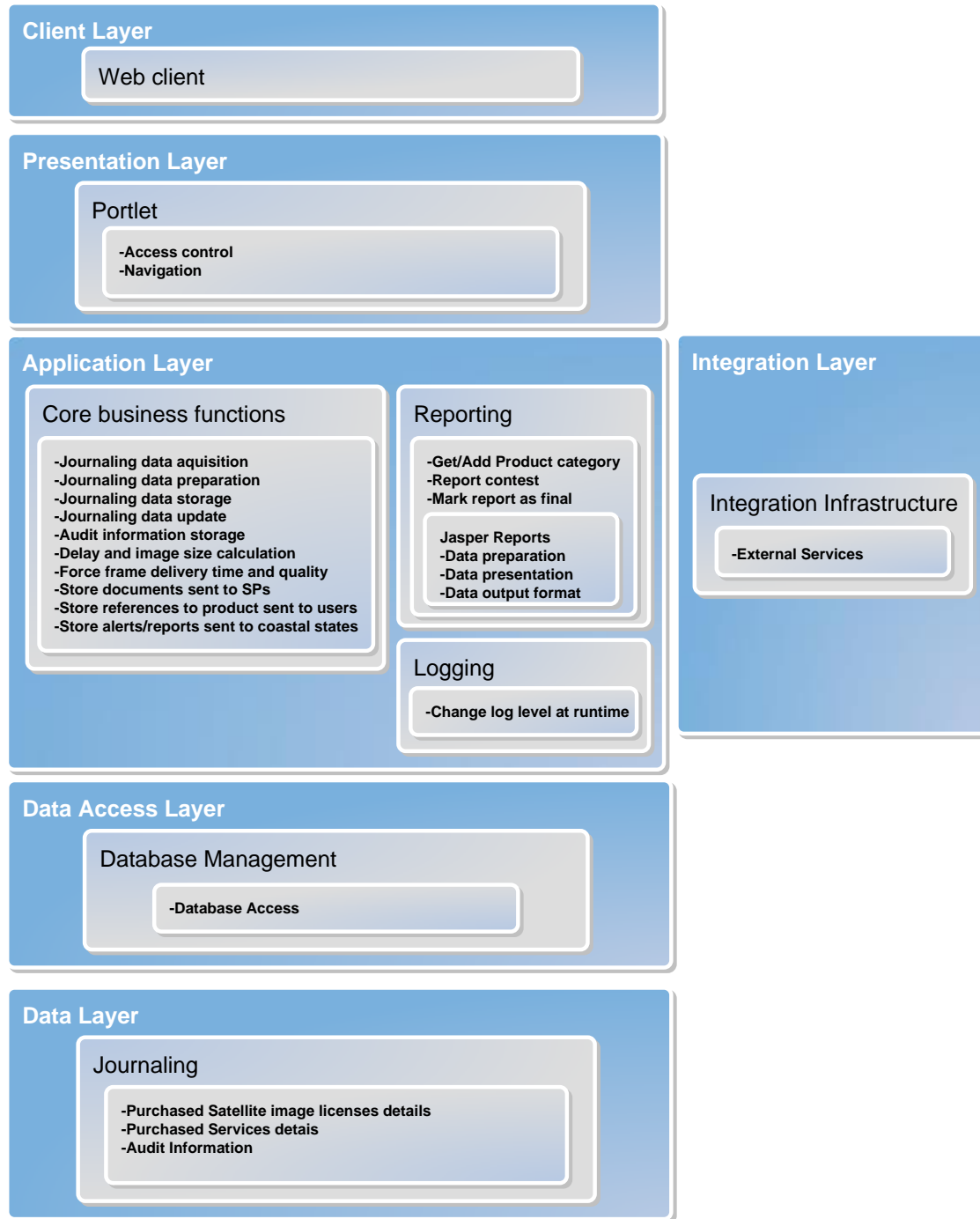


Figure 4-37: Journaling Logical View

4.9.1 Client Layer

The client layer represents thin clients, such as browser based web clients. The main objectives are:

- Handling of user input;
- Minimal validation capability on the client side;
- Transmission of end user input to the server;
- Receive reply from the server;
- Display received replies to the end user.

4.9.1.1 Web Client

Web Client	
Need:	<p>The Web Client component is a standard web browser such as Mozilla Firefox or Internet Explorer. This component communicates with the system via HTTP / HTTPS protocol, receives responses in HTML format and renders them for the user. The Web Client has general characteristics that include the following:</p> <ul style="list-style-type: none"> • Graphical Presentation • HTML Translation based on DTD • Applet Execution within a Java Virtual Machine (JVM) instance • JavaScript Execution • Plug-In Support • Caching • Security and encryption Services • Content Persistence (cookies)
Impact not having it:	Without the Web Client component no browser access to the JOU is possible.

4.9.2 Presentation Layer

4.9.2.1 Portlet

The Portlet component represents the server side capabilities that support the presentation of applications to thin clients. This application will reside on the Liferay portal component

4.9.2.1.1 Access control

Access control	
Need:	The access control component is responsible for managing the permissions associated to users, thus filtering what information is displayed to the user and what functionalities it has.
Impact not having it:	Without access control, there is no way to filter what information a user has access to. All users would access all information and functionalities.

4.9.2.1.2 Navigation

Navigation	
Need:	Navigation defines mechanisms to locate contents and applications without use of search functionality by navigating through a predefined structure. This structure can for instance organize portal services.
Impact not having it:	Every system needs navigation structure.

4.9.3 Application Layer

4.9.3.1 Core Business Functions

The core business functions component is responsible for all actions related to handling journaling information. These components will be installed on the back end web service, unless otherwise stated.

4.9.3.1.1 Journaling data acquisition

Journaling data acquisition	
Need:	The JOU must acquire all journaling data for further processing and storage.
Impact not having it:	In the case no data is acquired, no journaling information will be available through reporting.

4.9.3.1.2 Journaling data preparation

Journaling data preparation	
Need:	After acquiring journaling data, it has to be formatted and validated in order to be stored.
Impact not having it:	Stored data may not be valid or in a readable format.

4.9.3.1.3 Journaling data storage

Journaling data storage	
Need:	Journaling data has to be stored in order to be made available through reporting.
Impact not having it:	If no data is stored, there won't be information to display in reports.

4.9.3.1.4 Journaling data update

Journaling data update	
Need:	Journaling data may be updated by allowing the addition of new data as a complement.
Impact not having it:	Journaling data may not be complemented, and all stored data becomes final.

4.9.3.1.5 Audit information storage

Audit information storage	
Need:	The JOU must store all human-made data changes.
Impact not having it:	Data may be changed without knowledge of the user responsible.

4.9.3.1.6 Image delay calculation

Delay and image size calculation	
Need:	The JOU must calculate the values for image delay.
Impact not having it:	There is no way to tell the delay for any image, thus becoming impossible to provide these values to the external billing service.

4.9.3.1.7 Force frame delivery time and quality

Force frame class	
Need:	The JOU must allow the forcing of a class by setting manual delay and quality parameters.
Impact not having it:	After a class is determined there is no way to change it in case of need.

4.9.3.1.8 Store documents sent to SPs

Store documents sent to SPs	
Need:	The JOU must store all documents sent to SPs.

Impact not having it:	There is no way to tell what documents were sent to SPs.
------------------------------	--

4.9.3.1.9 Store references to products sent to users

Store references to products sent to users	
Need:	The JOU must store a reference to all products delivered to users.
Impact not having it:	There is no way to track what products were delivered to whom.

4.9.3.1.10 Store alerts/reports sent to coastal states

Store alerts/reports sent to coast states	
Need:	The JOU must store all the alerts and reports sent to coastal states.
Impact not having it:	There is no way to determine if an alert was sent or a report delivered.

4.9.3.2 Reporting

The reporting component is responsible for handling report requests and report handling, report exporting or report updating. The reporting components are installed on the liferay portal, as they are mainly related to the display of the information to the user. It must of course be noted that the information

4.9.3.2.1 Jasper Reports

In order to facilitate the presentation of reports, the Jasper Reports Library will be used to perform the following functionalities

4.9.3.2.1.1 Data preparation

Data preparation	
Need:	After a report request is received, data has to be acquired according to a set of report parameters. This data will be used to compose the report.
Impact not having it:	If no data is fetched from storage, then no data will be presented on the report.

4.9.3.2.1.2 Data presentation

Data presentation	
Need:	The data presentation is responsible for formatting the data in order to send it to the presentation layer to be displayed to the end user.
Impact not having it:	Data will not be formatted in a way compatible with the presentation layer, thus making it impossible to display a human readable report.

4.9.3.2.1.3 Data output format

Data output format	
Need:	The Data Output Format is responsible for the export of information in PDF, Microsoft Office Excel format or any other required format. The component isolation minimizes the impact of a possible future addition of new formats.
Impact not having it:	Without the Data Output Format component it will not be possible for the user to get the desired information in a different format.

4.9.3.2.2 Get/Add product category

Get/Add product category	
Need:	The JOU must allow to add categories to products and to get the categories of products.
Impact not having it:	There is no way to change or view the category of a product.

4.9.3.2.3 Report contest

Report contest	
Need:	In the case there of disagreement regarding the classification of a report, it must be possible to contest it.
Impact not having it:	By not being possible to contest, all classifications are final as soon as they are

4.9.3.2.4 Mark report as final

Mark report as final	
Need:	After a 48 hour period after the arrival of a report, the users must be able to mark the report as final, unless there is an active contest for it.

Impact not having it:	No report may be marked as final, meaning that changes can still be made to it.
------------------------------	---

4.9.3.3 Logging

The logging component is responsible for logging all operations, events and transactions occurring in the GUI JOU component. The main goal is to facilitate the tracking of problem causes and debugging.

4.9.3.3.1 Change log level at runtime

Change log level at runtime	
Need:	There is the need to change log level at runtime to facilitate the tracing of system errors.
Impact not having it:	Without being able to change the log level at runtime, the system had to be brought offline to change the log level.

4.9.4 Integration Layer

4.9.4.1 Integration Infrastructure

The Integration Infrastructure component is responsible for the integration of JOU with an external application. The integration will be performed through a web service using the Proxy paradigm, where no access is performed directly to the web service but through a proxy, in order to allow higher extensibility.

4.9.4.1.1 External Services

External Services	
Need:	This component enables the connection with external services.
Impact not having it:	Connection to external services becomes impossible.

4.9.5 Data Access Layer

Data Access Layer is responsible for the access to data in the data layer.

4.9.5.1 Database Management

4.9.5.1.1 Database access

Database access	
Need:	Provides access to databases. It will allow inserting, modifying and reading data from databases.
Impact not having it:	JOU databases would have no access, thus they would not be used at all.

4.9.6 Data Layer

4.9.6.1 Journaling

Journaling	
Need:	Component responsible for storing data flow information, satellite image license details and service details.
Impact not having it:	No data flow information, satellite image license details or service details would be available.

4.10 Communication Mechanisms (COM)

The overall functional component diagram, with the grouping of components into layers for the COM is depicted in the figure bellow. The sections that follow the mentioned figure describe those components. For this component all its sub components execute on the portal, and all data access is made through the portal's own DAL.

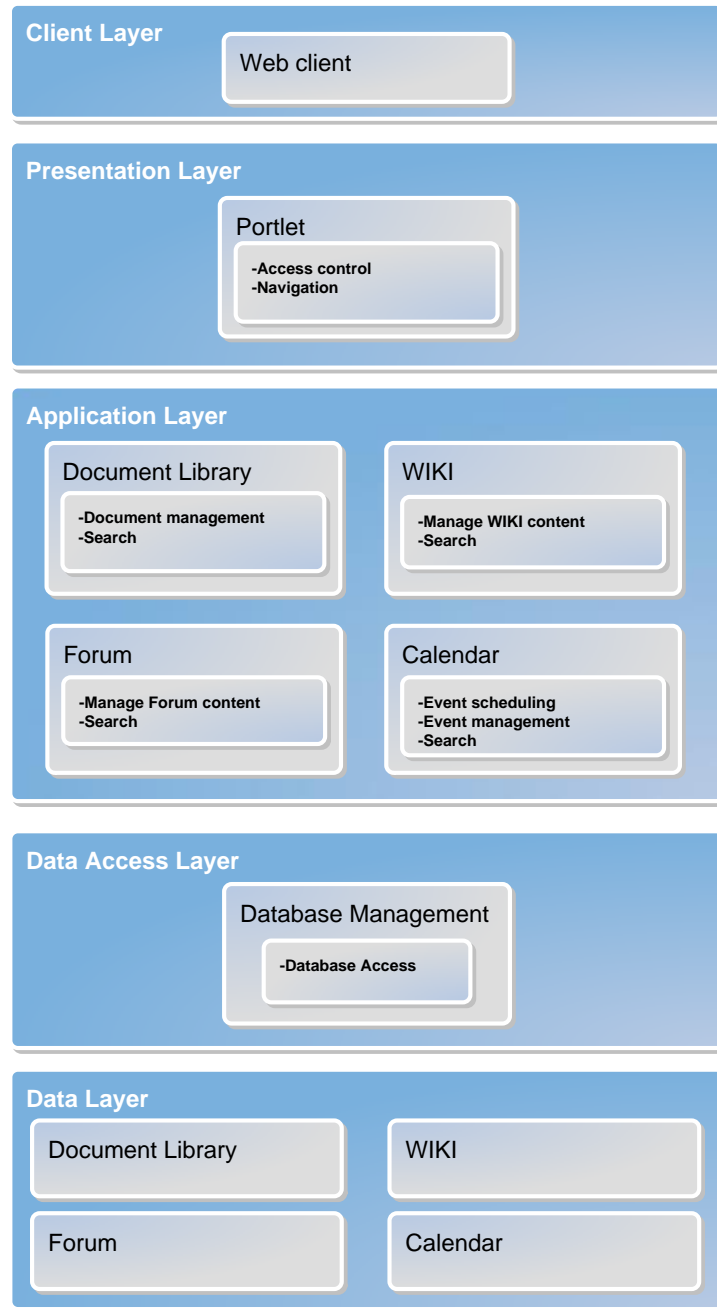


Figure 4-38: Communication Mechanisms Logical View

4.10.1 Client Layer

The client layer represents thin clients, such as browser based web clients. The main objectives are:

- Handling of user input;
- Minimal validation capability on the client side;
- Transmission of end user input to the server;
- Receive reply from the server;
- Display received replies to the end user.

4.10.1.1 Web Client

Web Client	
Need:	<p>The Web Client component is a standard web browser such as Mozilla Firefox or Internet Explorer. This component communicates with the system via HTTP / HTTPS protocol, receives responses in HTML format and renders them for the user. The Web Client has general characteristics that include the following:</p> <ul style="list-style-type: none"> • Graphical Presentation • HTML Translation based on DTD • Applet Execution within a Java Virtual Machine (JVM) instance • JavaScript Execution • Plug-In Support • Caching • Security and encryption Services • Content Persistence (cookies)
Impact not having it:	Without the Web Client component no browser access to the JOU is possible.

4.10.2 Presentation Layer

4.10.2.1 Portlet

The Portlet component represents the server side capabilities that support the presentation of applications to thin clients.

4.10.2.1.1 Access control

Access control	
Need:	The access control component is responsible for managing the permissions associated to users, thus filtering what information is displayed to the user and what functionalities it has.
Impact not having it:	Without access control, there is no way to filter what information a user has access to. All users would access all information and functionalities.

4.10.2.1.2 Navigation

Navigation	
Need:	Navigation defines mechanisms to locate contents and applications without use of search functionality by navigating through a predefined structure. This structure can for instance organize portal services.
Impact not having it:	Every system needs navigation structure.

4.10.3 Application Layer

4.10.3.1 Document Library

The document library component is responsible for handling document management (inserts, updates, deletes) and document search. This component is based on the Liferay Document library portlet and requires it to be present in order to execute properly.

4.10.3.1.1 Document management

Document management	
Need:	This component is responsible for all actions possible over documents and document information, such as: inserts/uploads, updates and deletes.
Impact not having it:	Without document management it becomes impossible for the end user to see and interact with documents in the document library.

4.10.3.1.2 Search

Search	
Need:	The search component is responsible for searching for documents according to a set of parameters selected by the user.
Impact not having it:	Without search, the user will not be able to filter documents according to a set of preferences and will have to navigate through the entire document library.

4.10.3.2 Forum

The forum component is responsible for handling forum management (post, edit and delete content) and content search.

4.10.3.2.1 Forum content management

Forum content management	
Need:	This component is responsible for all actions possible over forum content, such as: post, update and delete content.
Impact not having it:	Without forum content management it becomes impossible for the end user to see and create new forum content.

4.10.3.2.2 Search

Search	
Need:	The search component is responsible for searching for forum content according to a set of parameters selected by the user.
Impact not having it:	Without search, the user will not be able to filter content according to a set of preferences and will have to navigate through the entire forum.

4.10.3.3 WIKI

The WIKI component is responsible for handling WIKI management (post, edit and delete content) and content search. This component is based on the Liferay Wiki portlet and requires it to be present in order to execute properly.

4.10.3.3.1 WIKI content management

WIKI content management	
Need:	This component is responsible for all actions possible over WIKI content, such as: post, update and delete content.
Impact not having it:	Without WIKI content management it becomes impossible for the end user to see and create new WIKI content.

4.10.3.3.2 Search

Search	
Need:	The search component is responsible for searching for WIKI content according to a set of parameters selected by the user.
Impact not having it:	Without search, the user will not be able to filter content according to a set of preferences and will have to navigate through the entire WIKI.

4.10.3.4 Calendar

The Calendar component is responsible for handling event management (schedule, update, delete) and event search. This component is based on the Liferay Calendar portlet and requires it to be present in order to execute properly.

4.10.3.4.1 Event management

Event management	
Need:	This component is responsible for all actions possible over events, such as: schedule, update and delete events.
Impact not having it:	Without event management it becomes impossible for the end user to see and schedule new events.

4.10.3.4.2 Search

Search	
Need:	The search component is responsible for searching for forum content according to a set of parameters selected by the user.
Impact not having it:	Without search, the user will not be able to filter content according to a set of preferences and will have to navigate through the entire forum.

4.10.4 Data Access Layer

Data Access Layer is responsible for the access to data in the data layer.

4.10.4.1 Database Management

4.10.4.1.1 Database access

Database access	
Need:	Provides access to databases. It will allow inserting, modifying and reading data from databases.
Impact not having it:	JOU databases would have no access, thus they would not be used at all.

4.10.5 Data Layer

4.10.5.1 Document Library

Document library	
Need:	Component responsible for storing data related to documents and respective information in the COM databases.
Impact not having it:	No documents or document information would be available.

4.10.5.2 Forum

Forum	
Need:	Component responsible for storing forum related data in the COM databases.
Impact not having it:	No forum information would be available.

4.10.5.3 WIKI

WIKI	
-------------	--

Need:	Component responsible for storing WIKI related data in the COM databases.
Impact not having it:	No WIKI information would be available.

4.10.5.4 Calendar

Calendar	
Need:	Component responsible for storing calendar and event related data in the COM databases.
Impact not having it:	No calendar or events information would be available.

4.11 FinSys – Financial System

The overall functional component diagram, with the grouping of components into layers for the FinSys is depicted in the figure bellow. The sections that follow the mentioned figure describe those components.

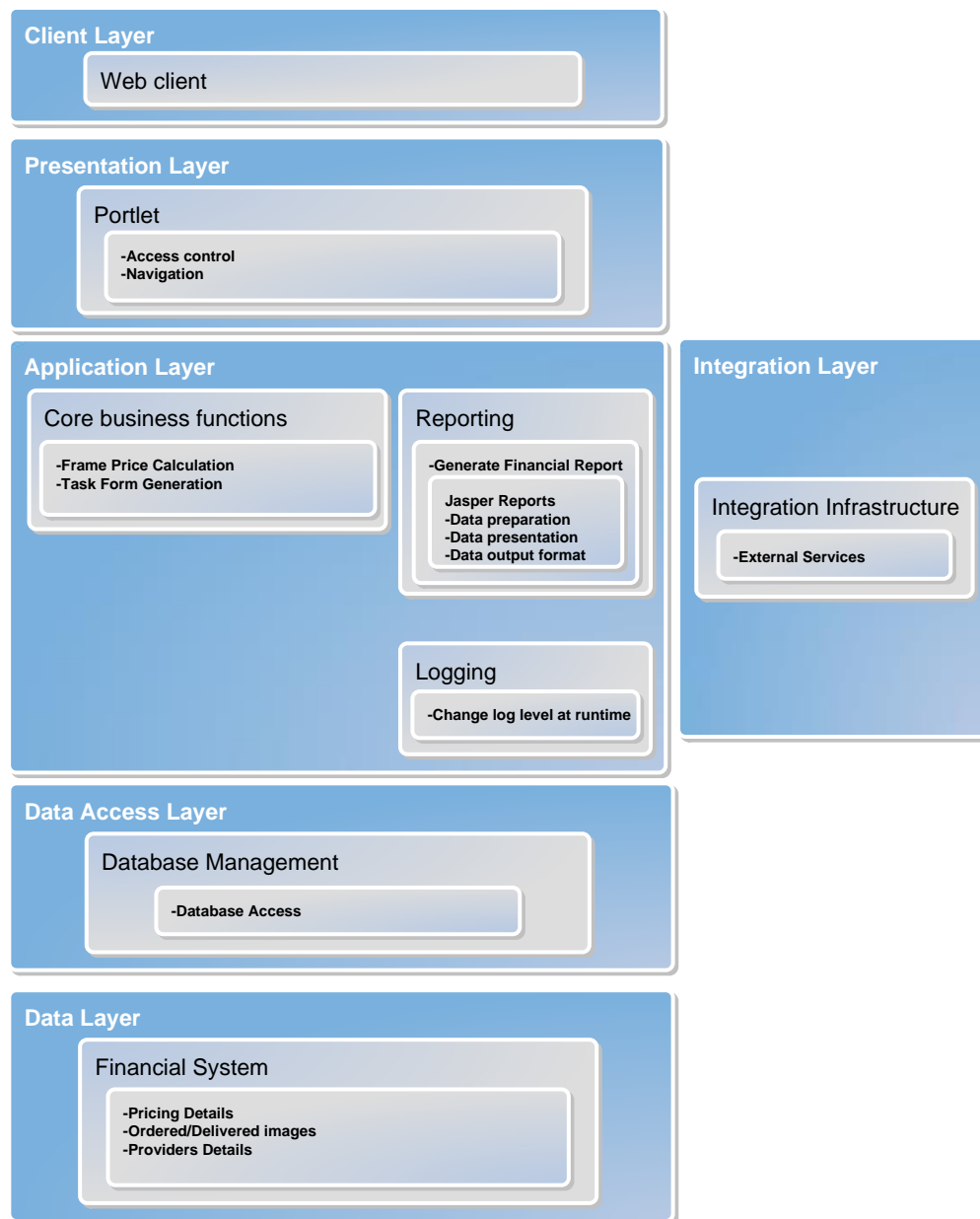


Figure 4-39: Journaling Logical View

4.11.1 Client Layer

The client layer represents thin clients, such as browser based web clients. The main objectives are:

- Handling of user input;
- Minimal validation capability on the client side;

- Transmission of end user input to the server;
- Receive reply from the server;
- Display received replies to the end user.

4.11.1.1 Web Client

Web Client	
Need:	<p>The Web Client component is a standard web browser such as Mozilla Firefox or Internet Explorer. This component communicates with the system via HTTP / HTTPS protocol, receives responses in HTML format and renders them for the user. The Web Client has general characteristics that include the following:</p> <ul style="list-style-type: none"> • Graphical Presentation • HTML Translation based on DTD • Applet Execution within a Java Virtual Machine (JVM) instance • JavaScript Execution • Plug-In Support • Caching • Security and encryption Services • Content Persistence (cookies)
Impact not having it:	Without the Web Client component no browser access to the JOU is possible.

4.11.2 Presentation Layer

4.11.2.1 Portlet

The Portlet component represents the server side capabilities that support the presentation of applications to thin clients. This application will reside on the Liferay portal component

4.11.2.1.1 Access control

Access control	
Need:	The access control component is responsible for managing the permissions associated to users, thus filtering what information is displayed to the user and what functionalities it has.
Impact not having it:	Without access control, there is no way to filter what information a user has access to. All users would access all information and functionalities.

4.11.2.1.2 Navigation

Navigation

Need:	Navigation defines mechanisms to locate contents and applications without use of search functionality by navigating through a predefined structure. This structure can for instance organize portal services.
Impact not having it:	Every system needs navigation structure.

4.11.3 Application Layer

4.11.3.1 Core Business Functions

The core business functions component is responsible for all actions related to handling journaling information. These components will be installed on the back end web service, unless otherwise stated.

4.11.3.1.1 Frame Price Calculation

Frame Price Calculation	
Need:	The finsys needs to calculate the individual price for each frame ordered and received
Impact not having it:	No price information is available.

4.11.3.1.2 Task Form Generation

Task Form Generation	
Need:	The finsys is responsible to generate the task forms to be sent to the providers
Impact not having it:	Task forms are not generated.

4.11.3.2 Reporting

The reporting component is responsible for handling report requests and report handling, report exporting or report updating. The reporting components are installed on the liferay portal, as they are mainly related to the display of the information to the user. It must of course be noted that the information

4.11.3.2.1 Jasper Reports

In order to facilitate the presentation of reports, the Jasper Reports Library will be used to perform the following functionalities

4.11.3.2.1.1 Data preparation

Data preparation	
Need:	After a report request is received, data has to be acquired according to a set of report parameters. This data will be used to compose the report.
Impact not having it:	If no data is fetched from storage, then no data will be presented on the report.

4.11.3.2.1.2 Data presentation

Data presentation	
Need:	The data presentation is responsible for formatting the data in order to send it to the presentation layer to be displayed to the end user.
Impact not having it:	Data will not be formatted in a way compatible with the presentation layer, thus making it impossible to display a human readable report.

4.11.3.2.1.3 Data output format

Data output format	
Need:	The Data Output Format is responsible for the export of information in PDF, Microsoft Office Excel format or any other required format. The component isolation minimizes the impact of a possible future addition of new formats.
Impact not having it:	Without the Data Output Format component it will not be possible for the user to get the desired information in a different format.

4.11.3.2.2 Generate Financial Report

Generate Financial Report	
Need:	The FinSys must allow the generation of financial reports
Impact not having it:	There is no way to generate the financial reports

4.11.3.3 Logging

The logging component is responsible for logging all operations, events and transactions occurring in the GUI JOU component. The main goal is to facilitate the tracking of problem causes and debugging.

4.11.3.3.1 Change log level at runtime

Change log level at runtime	
Need:	There is the need to change log level at runtime to facilitate the tracing of system errors.
Impact not having it:	Without being able to change the log level at runtime, the system had to be brought offline to change the log level.

4.11.4 Integration Layer

4.11.4.1 Integration Infrastructure

The Integration Infrastructure component is responsible for the integration of JOU with an external application. The integration will be performed through a web service using the Proxy paradigm, where no access is performed directly to the web service but through a proxy, in order to allow higher extensibility.

4.11.4.1.1 External Services

External Services	
Need:	This component enables the connection with external services.
Impact not having it:	Connection to external services becomes impossible.

4.11.5 Data Access Layer

Data Access Layer is responsible for the access to data in the data layer.

4.11.5.1 Database Management

4.11.5.1.1 Database access

Database access	
Need:	Provides access to databases. It will allow inserting, modifying and reading data from databases.
Impact not having it:	FinSys databases would have no access, thus they would not be used at all.

4.11.6 Data Layer

4.11.6.1 Financial System

Journaling	
Need:	Component responsible for storing data relating to the financial system.
Impact not having it:	No data is stored by the financial system and none of its functionalities are usable.

5 Technical Implementation View

This chapter reports the dynamic behaviour of the system, illustrating the most important use cases using sequence diagrams, that explain in detail the interfaces between the CSN-DC internal components. The chapter is logically divided into sections related to the system components, but in fact all the scenarios illustrated here are transversal to many components. Therefore even if the sequence diagrams are collocated in the section belonging to the component which is most relevant to the topic illustrated by the graph, they inherently belong to more than one section/component. As a consequence, this chapter shall be analysed “as a whole”.

5.1 DAM - Data Access Management

The following section reports some analysis of the dynamic model of the DAM. In this context the reference will be made to the DAM as a whole (and not to its internal modules) as this analysis shall focus on the overall CSN DC internal interfaces.

In this first example, the main high-level ingestion scenario is depicted.

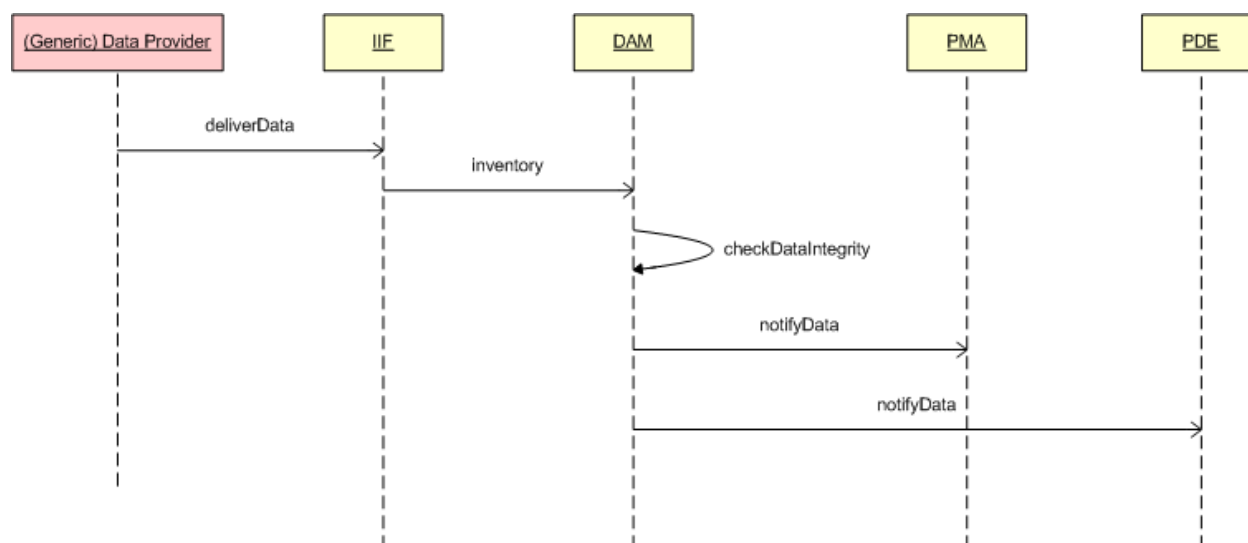


Figure 5-1 - DAM main high level ingestion scenario

The followings steps are executed:

1. The IIF retrieves data from the a generic data provider (e.g. oil spill analysis from the Service Providers)
2. The retrieved data is ingested into the DAM
3. The DAM checks the data integrity
4. After successful ingestion, a message is sent to the PMA and to the PDE

The PMA and the PDE may trigger execution of other functions on the basis of the arrival of the data. This will be further illustrated in the PMA section (§ 5.4) and the PDE section (§ 5.6).

In particular when data are received from the SPs, the process is slightly more complex, because the DAM shall also notify the JOU and the POR. This is illustrated in § 5.2.

In the following example, the WUP requests metadata (e.g. through CSW service) to the webcat:

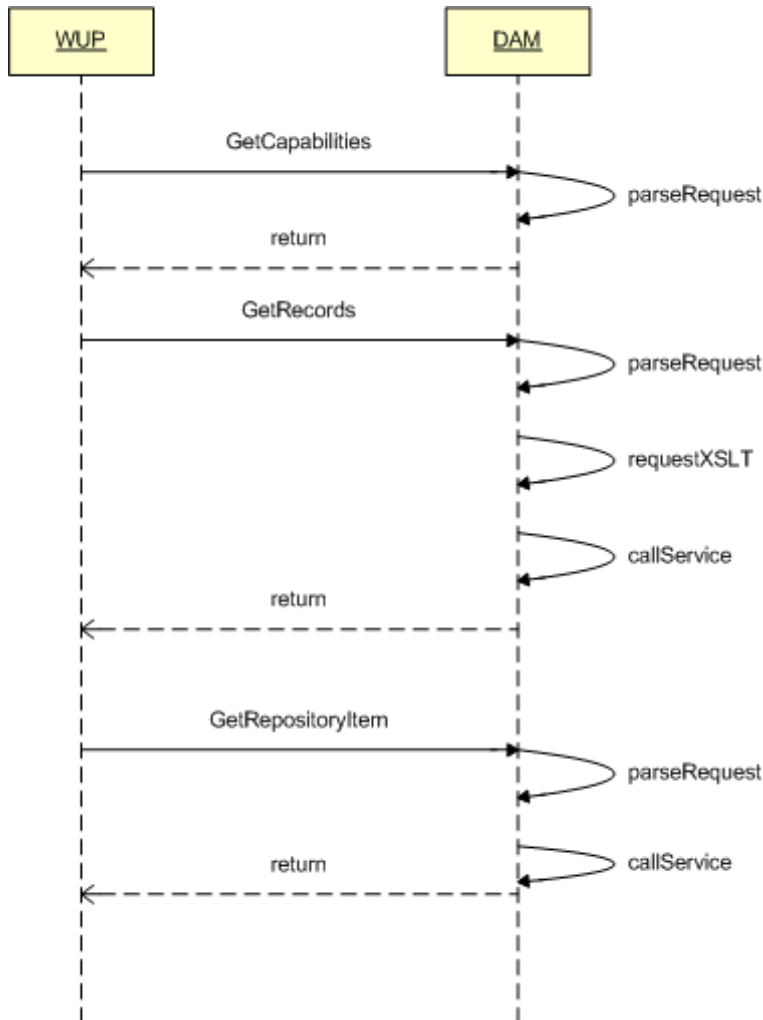


Figure 5-2 - WUP <-> DAM sequence diagram

The followings steps are executed:

1. The WUP rich client performs a preliminary GetCapabilities request to the Webcat service
2. The WUP rich client create and submit a GetRecords request
3. DAM Webcat module parses the request, translates incoming request to internal data representation (if needed) and call CSW Service. Output will be a number of CSW records (in GML/ebRIM format)
4. The WUP rich client performs a GetRepositoryItem to retrieve a specific full repository item (in GML format)

In the next example, DAM services are used by PDE to collect information and data to be included in an Alert. For the sake of simplicity, the example refers to the creation of an Alert in case of an Oil Spill detection and assumes that target user is interested in having information about the Oil Spill itself, about the vessel detected in the same EO scene in which the Oil Spill has been observed and

about wind meteo conditions in the same area. Please note: the focus of this sequence diagram is not the PDE internal logic but the way DAM services satisfy PDE data collection needs, so no details are reported here about the triggering of PDE activities and the delivery of the generated Alert.

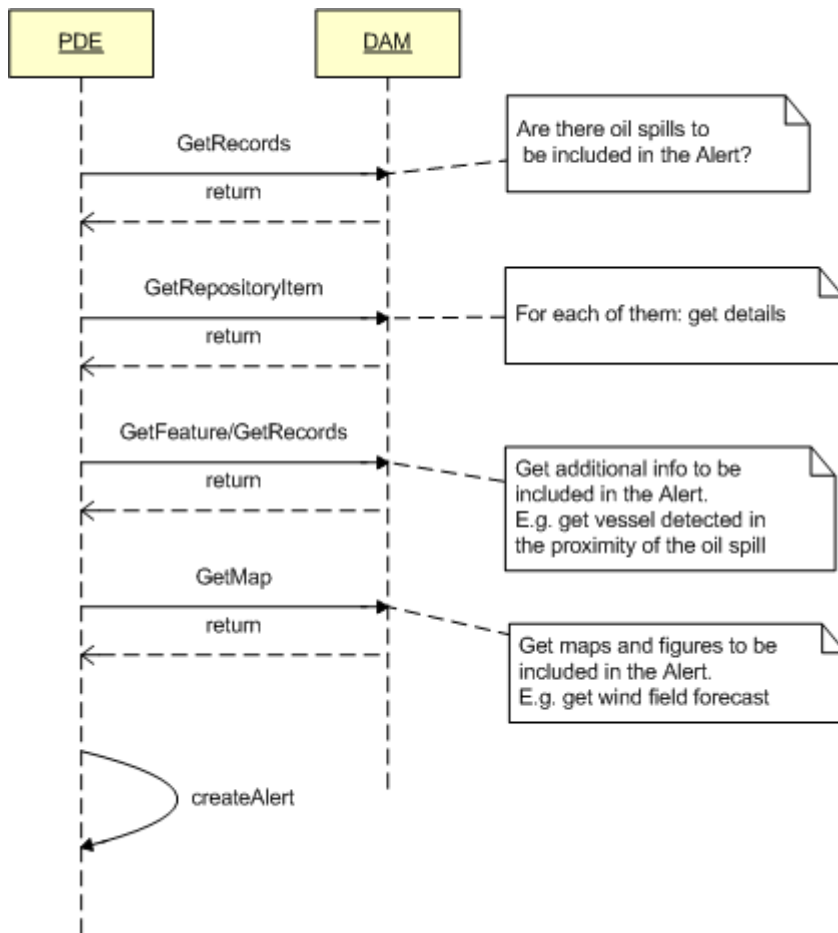


Figure 5-3 - Info collection for Alert creation sequence

The followings steps are executed:

1. The PDE uses the GetRecords call of the DAM's CSW to retrieve newly detected oil spills
2. For each oil spill, PDE performs a GetRepositoryItem call to retrieve metadata and details of the spill
3. Additional GetRecords or WFS GetFeature calls are performed to retrieve other information to be inserted in the Alert (for example the presence and the details of vessels detected in the same EO scene in which the oil spill has been observed)
4. GetMap calls to DAM's WMS can be performed to retrieve images to be inserted in the PDF Alert. For example: coastlines and geographic features, wind field,...

5.2 IIF - Ingestion and interfaces

The IIF is responsible for the interfaces with the external entities. The generic scenario for provision of data to the IIF has already been illustrated in the previous section (see Figure 5-1). This may correspond to a generic data provision interface, such as the MyOcean interface (which will be routinely sending data to the CSN-DC) or the other EO data provider interface, to which the CSN-DC may request data on an ad-hoc basis.

A special case shall be made for the SP, one of the most critical interfaces of the CSN-DC, where a slightly more complex scenario is foreseen and will be illustrated hereafter.

The SPs routinely deliver a series of data packages to the CSN-DC. The logic of data packages exchange has been illustrated in the [FDD] document and will not be repeated here in detail. Just to summarise, the main points are:

- SPs acquire and process SAR data and produce a number of outputs that shall be delivered to the CSN-DC. These outputs are distributed over a series of different packages which may be distributed asynchronously and include:
 - Oil spill warning sub-package
 - SAR Image sub-package
 - Oil spill notification sub-package
 - Quality sub-package
- All these sub-packages are logically linked by a unique order ID, which is a one-to-one link to a SAR scene ordered and processed
- The timeliness of data arrival is fundamental for verifying the performance of the SP (and consequently invoicing)
- The POR needs to be updated by setting the status flag of acquired scenes (to update the status of scenes which have been ordered)
- The JOU needs to keep track of the data delivery, including timeliness information, necessary for defining the invoicing

In order to account for data transmission delay which depends on the network bandwidth and therefore is not under the control of the SP, the following approach is devised:

1. Sub-packages belonging to the same SAR scene can be delivered independently on each other (they share a common order ID)
2. Before transmission of each sub-package, the SP sends a message of init delivery to the IIF, which includes:
 - a. MD5 digest of the sub-package
 - b. Order ID
3. The IIF notifies the init transmission to the JOU, referencing the order ID and the init delivery time
4. The IIF returns an acknowledge of receipt to the SP
5. The SP initiates the actual delivery of the data sub-package
6. When data is successfully received the IIF:
 - a. notifies the JOU indicating the time of arrival of the package
 - b. checks the digest versus the received sub-package
 - c. ingests the data into the DAM
7. The DAM checks the data integrity
8. The DAM updates the orders status on the POR
9. The DAM notifies the transmission to the JOU, indicating the order ID and the sensing time of the received SAR data

The init delivery message is sent from the SP to the IIF via a web service made available by the IIF. The order ID shall be used for uniquely identifying the SAR scene processed and will be used for updating the status of the POR and of the JOU. The time of init delivery is immediately

communicated to the JOU, along with the order ID. At the end of the processing, after the check of the digest versus the data package actually delivered (performed by the IIF) and the check of the data integrity (performed by the DAM), the JOU receives another message with the order ID (used to correlate) and the sensing time of the SAR image contained in the data package. By comparing the SAR sensing time with the end delivery time, the JOU shall be able to computing the timeliness of the service provision by the SPs. As already indicated above, this mechanism will also allow the POR to update the status flag of the SAR images ordered.

The bandwidth provided by EMSA should be sufficiently large to allow rapid transfer of the SP packages. In any case, the difference between start transmission and end transmission times shall be recorded into the JOU in order to be able to identify possible network problems. Likewise, in case the MD5 does not match with the received package, the JOU shall be notified and a detailed verification of the transmission process shall be initiated, to identify possible data corruption causes, e.g. related to the network.

In fact the POR orders SAR scenes and processing services related to the scenes, but the loop on the POR can only be closed when the SAR data and associated processing services are delivered by the SPs, with the order ID information.

The above process is illustrated in the following sequence diagram. Please note that, although a certain number of parameters are passed into the message calls, only the most important ones necessary to support understanding are explicitly illustrated. For example the order ID is extracted from a XML file passed on the web service call, which contains much more information, etc..

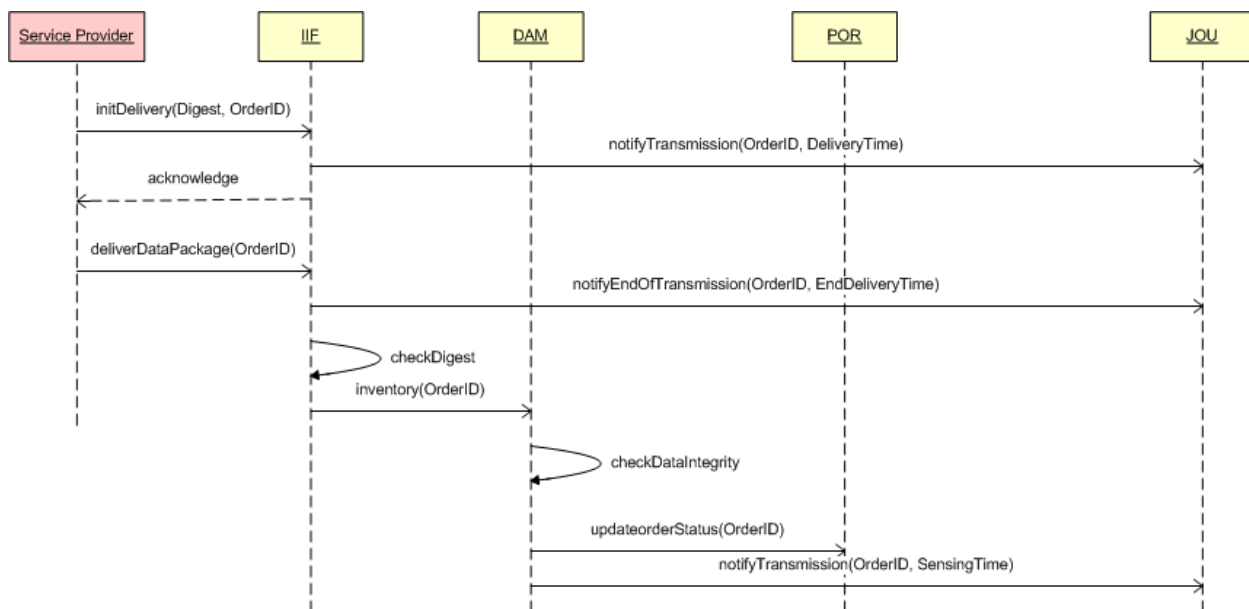


Figure 5-4 Mechanism for receiving the data packages from the SPs

This scenario, along with the generic one illustrated in the previous section covers most of the activities of the IIF. The following cases, that deeply involve other system components will be illustrated in the next sections:

- For each SAR scene processed, the IIF retrieves the corresponding vessel traffic data and makes them available in the CSN-DC for further analysis (see § 5.4, in particular Figure 5-19)

- Upon subscription or interactive request, the PDE instructs the IIF to deliver data to the user that have requested the data (see § 5.6)

A more detailed representation of the data flows for ingestion is depicted using the BPM notation, as in the following figure.

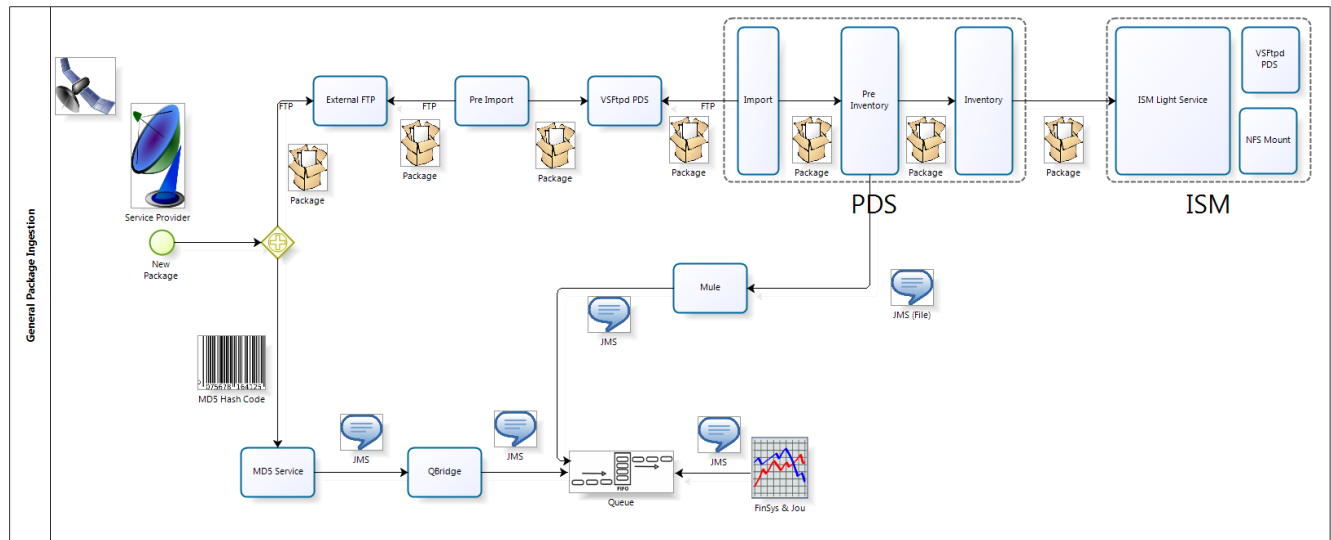


Figure 5-5 - Ingestion chain workflow

On the bottom line there is the MD5 service which simply announces that a given service is going to be disseminated. The ingestion component creates a JMS message that the then put on a JMS queue by a component named Qbridge. Finally the message is delivered to the JOU/FINSYS. When a product package is disseminated to the CSNDC this is loaded via the External FTP component, which retrieves the package from an FTP server (named External because it can be accessed externally) and is transferred internally by the PreImport. After this stage the PDS Import component is activated which performs pre-inventory (extraction of relevant metadata) and inventory (storing the metadata in the catalogue. The pre-inventory also creates a JMS message signalling the arrival of the product to the JOU. The final package is then stored into the ISM light storage for further access.

Details of the MD5 messages exchange between the service providers and the MD5 service (also referred to as Hash Service) are included in [EXT-IF-ICD], which reports a sequence diagram and the detailed content of all messages.

The interaction between the Ingestion Interface and the JOU described here is part of a more global interaction between components which entails also the POR and the FINSYS. The detailed dynamic behaviour of the messages exchanged between POR and JOU/FINSYS is described in §5.7.3, while here below a component diagram is reported which describes the dependencies between IIF, JOU, POR and FINSYS.

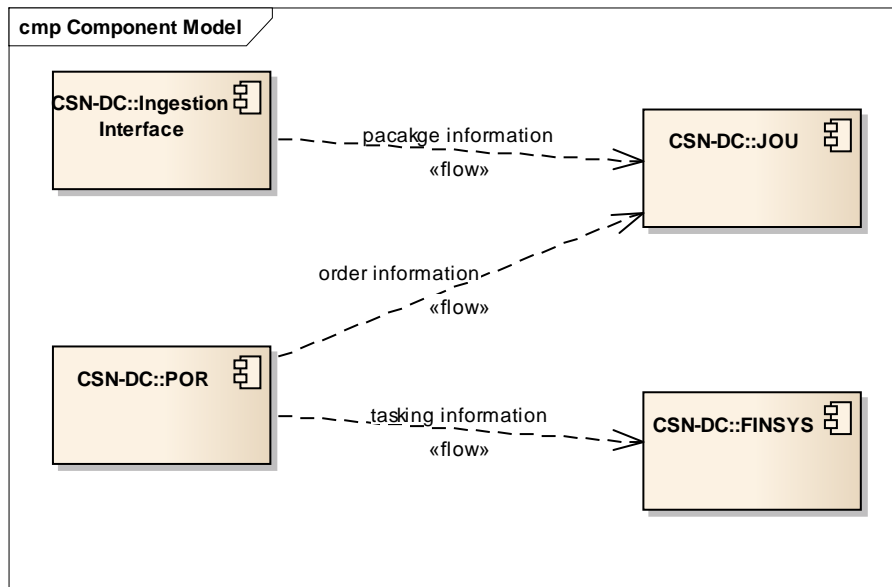


Figure 5-6 – Component diagram focussing on the dependencies between IIF, JOU, POR and FINSYS

The following diagram illustrates the workflow executed when EO scene is ingested. Even if this is the section dedicated to the IIF, this diagram contains also elements of the other components, e.g. the PMA (for data processing) and the DAM (for data storage).

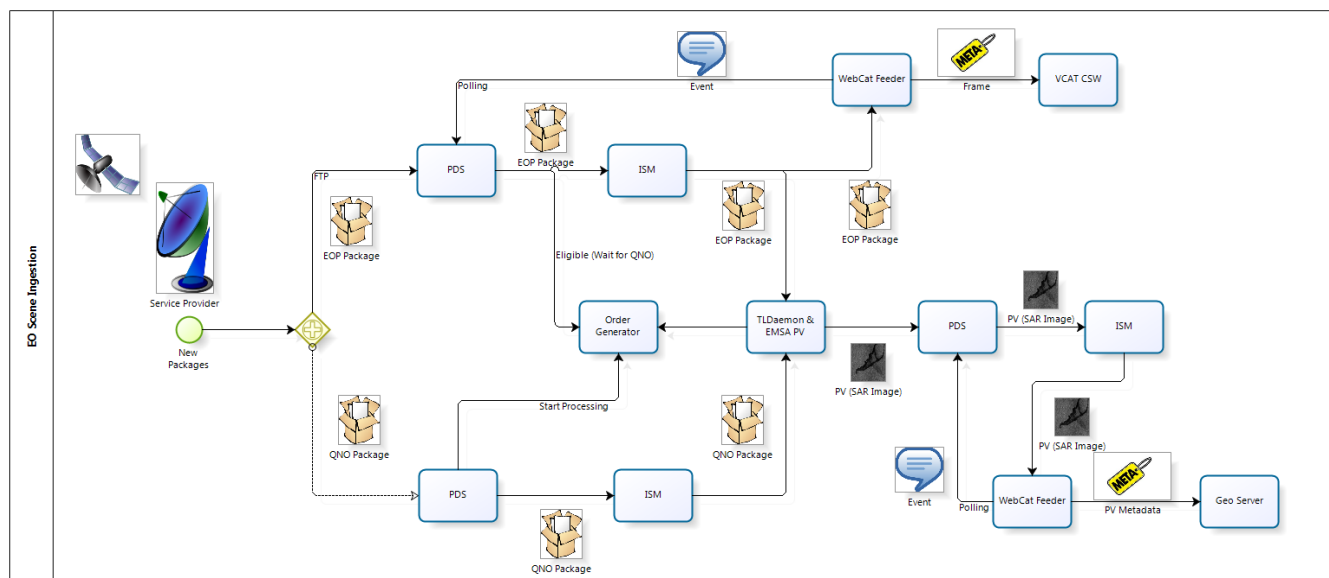


Figure 5-7 - Workflow for managing ingestion of a EOP (and QNO)

The workflow covers the ingestion of EOP package (and the QNO). In order to trigger the processing of the EOP data, it is necessary that both the EOP and the QNO area received, because the QNO may contain some data that are to be used on the processor.

If both are arrived, the Order Generator triggers a processing via the TLDaemon, which generates a file named EMSA PV (pyramidal view). Despite the name, this processor is not only creating a raster

file that uses a pyramidal representation paradigm for optimising the display at various zooming levels, but also performs some geometric and radiometric correction on the incoming SAR data (details are out of scope in this architectural document). The PV is a GeoTIFF file representing the raster data that shall be eventually displayed on the GIS Viewer. This file is handled by the WebCatFeeder which exploits the Geoserver REST APIs for ingesting the file. Basically the feeder creates the appropriate store and layer in geoserver and associated the GeoTIFF file to it. Once this is done, the file is ready to be accessed via the GIS viewer.

The next diagram illustrates a Component Diagram with interfaces of the WebCatFeeder, a major subcomponent of the Ingestion component, which stores data into the various COTS used for delivering OGC services.

In particular:

- store TIFF data into Geoserver
- store WFS Oil Spill, Detected vessels and AIS data, to the Deegree WFS
- store EOP metadata into the Deegree VCAT

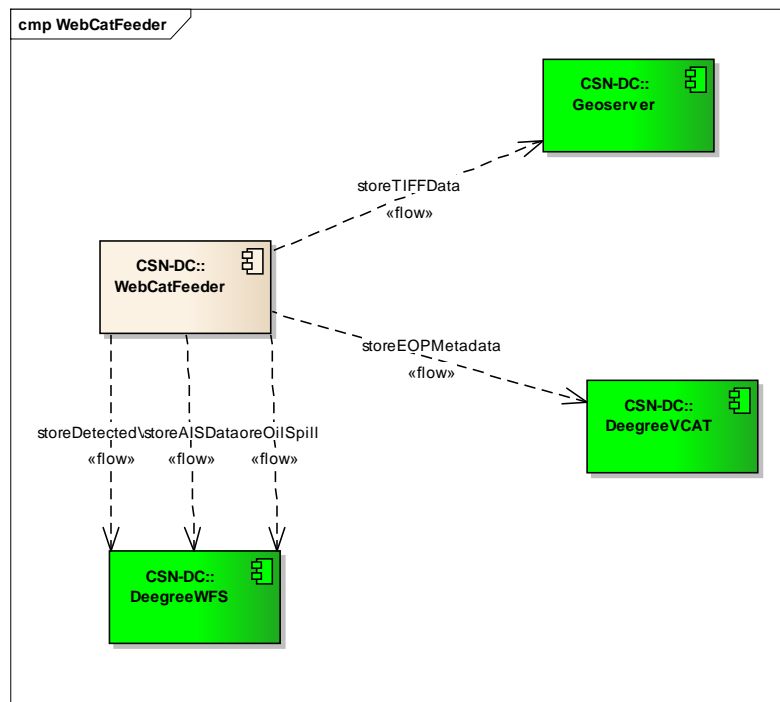


Figure 5-8 – WebCatFeeder interfaces with COTS

5.2.1 Ingestion of AIS data

An important component of the IIF is the ingestion of the vessel traffic data, which is done through I/F with IMDatE system. IMDatE is an integrated environment available at EMSA which collects all different sources of maritime traffic data (e.g. ship positions) from all available sources, thus including AIS, Satellite AIS, Long Range Identification and tracking (LRIT), Vessel Monitoring Systems (VMS), vessel Detection Systems (VDS), coastal radar, etc.

IMDatE offers a number of external I/Fs to access its data, out of which the following are particularly useful for the tasks of retrieving vessel traffic data in CSNDC:

- *Distribution service*: this is a REST web service I/F
- *Track service*: this is a SOAP web service I/F

The IMDatE distribution service works as a *subscription*. An external system can subscribe to IMDatE specifying the following parameters:

- Data type (e.g. shp position)
- Filter on the data type. This filter can be expressed using XQuery mechanism, thus allowing the maximum flexibility. In practise, the following criteria are used by CSNDC:
 - Data source (e.g. AIS, LRIT. Etc.)
 - Area bounding box (the bounding box of the EO scene for which the vessel traffic data shall be retrieved)
- Format, e.g. XML, csv
- Aggregation strategy (data are distributed in packages, this parameter defines how many data objects are collected before a distribution file is packaged and sent to CSNDC)
- Distribution mechanism, e.g. FTP, email

Once a distribution is configured, IMDatE starts distributing the data matching the condition expressed in the distribution, using the means as configured (e.g. channel, aggregation strategy, format, end point addressed, etc.). In the CSNDC typical real case, data are distributed via FTP on machine and folder that are predetermined by CSNDC. As part of the CSNDC vessel traffic ingestion engine, it is possible to configure the following:

- FTP end point and credentials
- *Base Folder* where to place the FTP data
- Aggregation strategy

CSNDC will automatically define a subfolder, below the base folder, which is dedicated to the specific service ID, thus includes the service ID in the name. This will enforce separation of data for different services.

CSNDC needs to collect the vessel traffic data for the 6 hours before the corresponding EO images sensing time up to 6 minutes after the sensing stop time. These time ranges are nominal cases and are in any case configurable in the CSNDC side.

The vessel collection engine, when it reaches 6 hours before acquisition of a given scene, creates a new subscription on IMDatE and then routinely (e.g. every 6 minutes) starts collecting the data that are distributed by CSNDC via FTP as a result of the subscription. When the time expires (typically > 6 minutes after the EO sensing stop), CSNDC closes the subscription, by deleting its corresponding record in the IMDatE and stops collecting data from the FTP endpoint.

While harvesting data in the FTP endpoint, CSNDC performs the following processing tasks:

- Ingests data inside the CSNDC database
- Transforms data into the GML format and stores it into another storage endpoint (typically an FTP destination, that shall be eventually accessed by external users, e.g. Service Providers)

GML data distributed via file are in chunks whose size depends on the aggregation factor.

NOTE: the size of the individual files is \geq aggregation factor. IMDatE uses the aggregation factor as a lower limit before sending the file. Since IMDatE checks the aggregation criteria every few seconds,

and since the data to be aggregated are collected in real time, it may happen that the aggregated chunk is slightly bigger than the minimum size (1-2 more data points).

The *Track Service* is used to complete the vessel tracks for those vessels that exited the EOP bounding box area before the sensing time. This step is necessary because since the subscription only delivers data that are within the bounding box defined by CSNDC, a vessel existing this area before the sensing time, will not be represented by the whole track covering the time of interest. The Track service allows to query vessel traffic data for a specific ship and time range. The general mechanism for calling the track service is the following, and is performed when the subscription expires (see above):

- For each distinct vessel obtained during the distribution get max position time
 - If *EOP sensing stop time - max position time > TIME_THRESHOLD* THEN get additional data via track service (the time range for the track service request would be: from max time to EOP sensing stop time + 6 minutes)

Data retrieved from the track service request will simply complete the incomplete tracks for those vessels, in practice they are treated in the same manner as the data obtained from the distribution services, e.g.:

- Stored in the CSNDC database
- Converted into GML files and exposed on FTP

The size of the GML file created after calling the Track Service is variable, depending on the number of ships tracks to be extended and amount of data needed for extending all tracks. Typically this is considerably bigger than the chunks obtained by each individual distribution (a typical case could include 1000-2000 points, but it depends on the vessel traffic scenario).

Examples of the GML files produced are reported in the [EXT-IF-ICD].

The complete sequence of actions performed by the Vessel traffic ingestion engine is reported in the following diagram. The sequence is executed routinely and triggered by a cron job (nominally every 6 minutes). The following steps are performed:

- Vessel Traffic Engine retrieves the list of eligible scenes from the POR database. A scene is eligible if the EO sensing start time is $< \text{NOW(UTC)} + 6 \text{ hours}$
- For each eligible scene:
 - If the distribution was not already created then create a distribution
 - Harvest distributed data (in case the distribution was created during the previous run of the engine, there will probably already be distributed file to be collected)
 - Store harvested vessel traffic data in the internal database
 - Create GML files
 - Store GML files in the External FTP server
- Vessel Traffic Engine retrieves the list of expired scenes from the POR database. A scene is expired if the EO sensing stop time is $< \text{NOW(UTC)} - 6 \text{ minutes}$
- For each expired scene:
 - Find the list of incomplete vessels (those for which the max time is less than the EO sensing stop – configurable threshold)
 - For each incomplete vessel:
 - Get track by calling the Track Service with the following parameters:
 - MMSI of the vessel
 - Max time of the available track
 - EOP sensing stop + 6 minutes

- Store the vessel traffic information obtained by performing the various vessel track requests
- Create corresponding GML file (a unique file including all missing pieces of the incomplete vessels)
- Store GML files on the External FTP server
- Delete the distribution from IMDatE

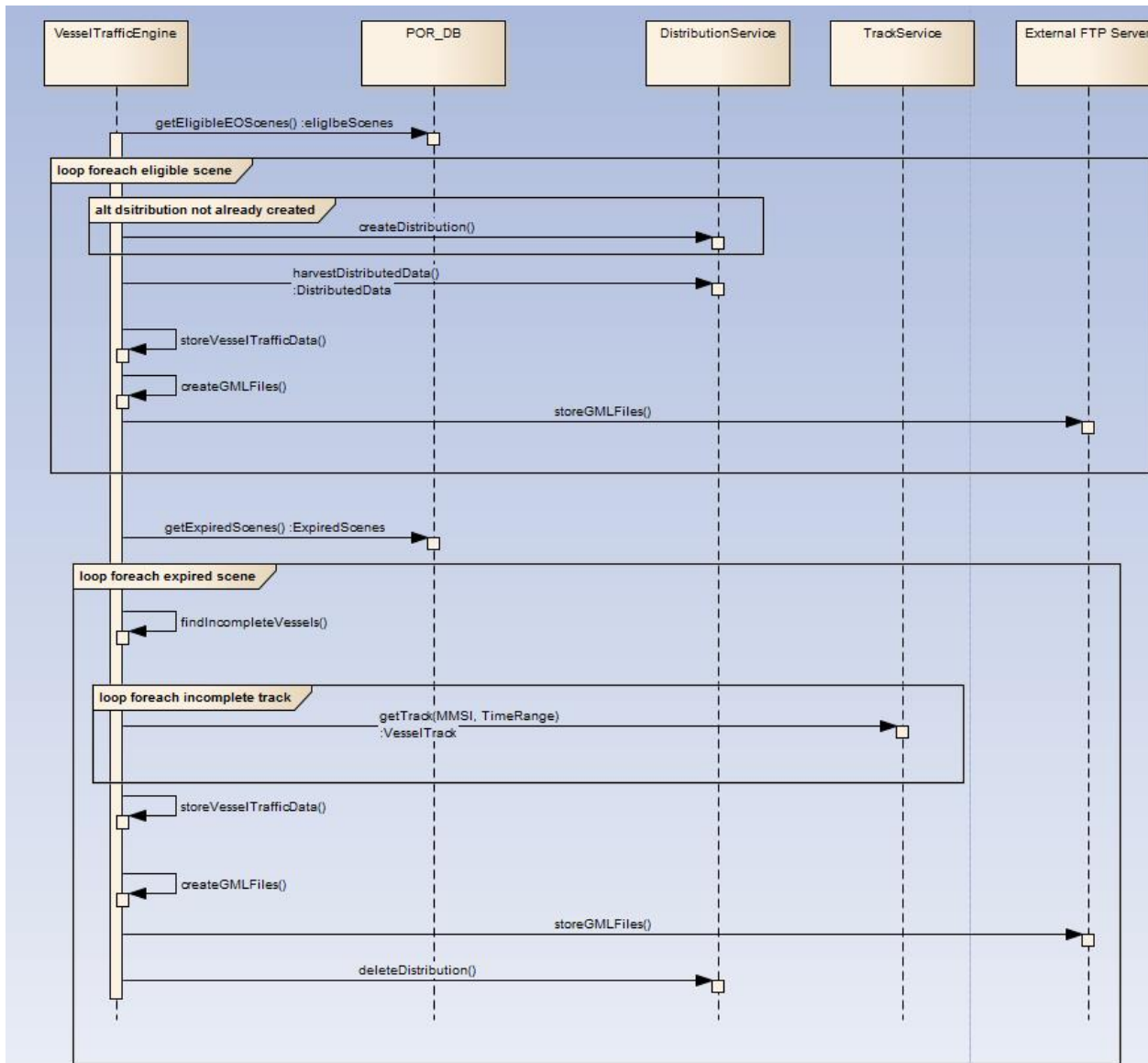


Figure 5-9 – Vessel Traffic information retrieval sequence diagram

5.2.2 Notification service

One of the features of the ingestion chain is to create notification messages following the SES OGC specifications to be possibly used by other systems, e.g. to arrange workflows to access to data made available by CSN-DC.

The overall concept is based on a notifier-subscriber pattern, with the following main elements:

- the *NotificationProducer*
- the Event Service
- the subscriber

In this scenario, the *NotificationProducer* is CSN-DC, while the Event Service and the subscriber are other project, e.g. the IMDatE.

Therefore the CSN-DC implementation basically consists in implementing a notification system, which is in line with the specification of the Event Service. In particular the **52North** COTS was used as a reference (<http://52north.org/>), as being the only reliable COTS available around which implements the SES OGS specification (<http://www.opengeospatial.org/resource/products/byspec>).

The notification message produced by CSN-DC is compliant with the OGC SES and in particular with the O&M specification.

In particular the details of the message have been defined taking into consideration the filtering capabilities of the **52North**. For example the COTS, even if suggests that the polygon filtering for AOI is supported, for the time being only supports AOI filtering based on points. Therefore, even if the event Service is not part of this procurement, the message details have been tuned for this target. In the future, the message format can be easily adjusted, as it is based on the application of an appropriate stylesheet.

The CSN-DC implementation is basically consisting in a notification message which is generated for each data package received. The complete specification of the messages and examples are documented in [EXT-IF-ICD].

NOTE: while this service was designed in order to be standard and to be compliant with the most reliable Event Service COTS, the architecture is **fully** compliant with any other solution, which does not use the Event Service. In fact the notification message is a simple SOAP call, which can be managed by any SOAP listener (which could be, e.g.: an Event Service COTS, a listener implemented using OSB, or any other simple or complex bespoke listener mechanism). In fact it is demonstrated in the test cases below, that the test is performed without the Event Service COTS, using a simple event listener stub program which is in a configurable location.

The notification message contains relevant information about the ingested data, and also the pointer to make a full request for the data, thus covering all possible use cases.

The message syntax is validated using any XML analysis tool (e.g. XML Spy) and validating it against the xsd. Regarding the information included in the message, the following is expected:

- timePosition : start acquisition time of the service ingested
- featureOfInterest: a OGC call that allows to retrieve all details of the ingested data, for example the reference to the oil spill which generated the event notification. This string is HTML encoded, therefore it shall be HTML decoded before submitting the CSW request.
- SamplingPoint: ID of the feature (e.g. internal ID of the oil spill)
- Name: name of the package which was ingested

- Point: centroid of the feature (e.g. centroid of the oil spill to which the event refers)
- om:observedProperty: feature type (it can be: OilSpill, EOscene, Vessel)

A component diagram illustrating the dependencies is reported below. It is only composed by the WebCatFeeder CSN-DC and the Sensor Event Service, which is an element external to the CSN-DC procurement (e.g. a COTS). As a matter of fact nothing prevents to implement the receiver of the messages not as an event service component (which deliver a number of specific functionalities, such as subscription, etc.), but simply as a listener which upon reception of a given message (SOAP in this case) will perform some specific workflow (e.g. this could be implemented by the EMSA OSB).

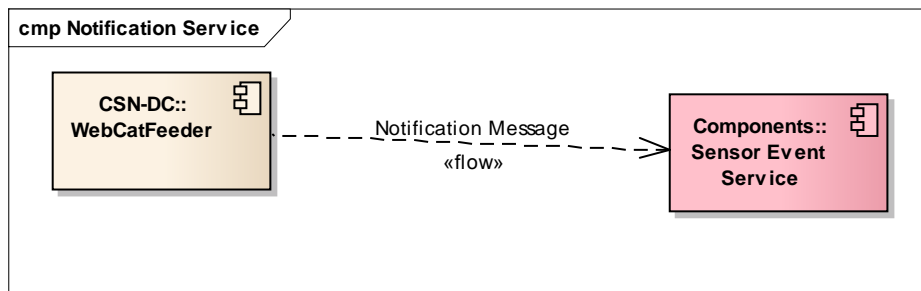


Figure 5-10 – Notification Service Component Diagram

Here follows a sequence diagram with the description of the information flow. The schema is very simple:

- ingestion interface receives a new data package
- ingestion interface ingests the new package
- upon successful ingestion of the package the ingestion interface sends a notification messages to the configured endpoint

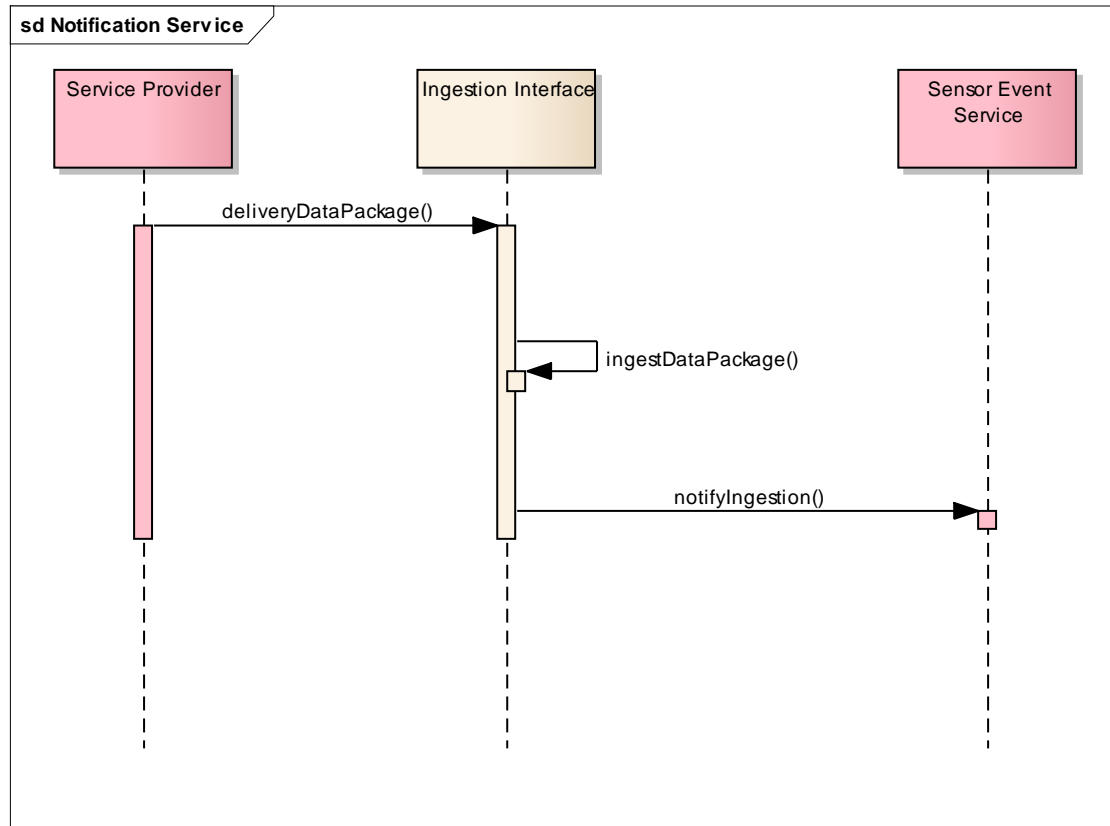


Figure 5-11 – Notification Service sequence diagram

5.3 UMA - User Management

This section reports the implementation details of the UMA component. The overview of this functionality is reported in § 4.4.1.

A component model of the UM is reported in the following diagram.

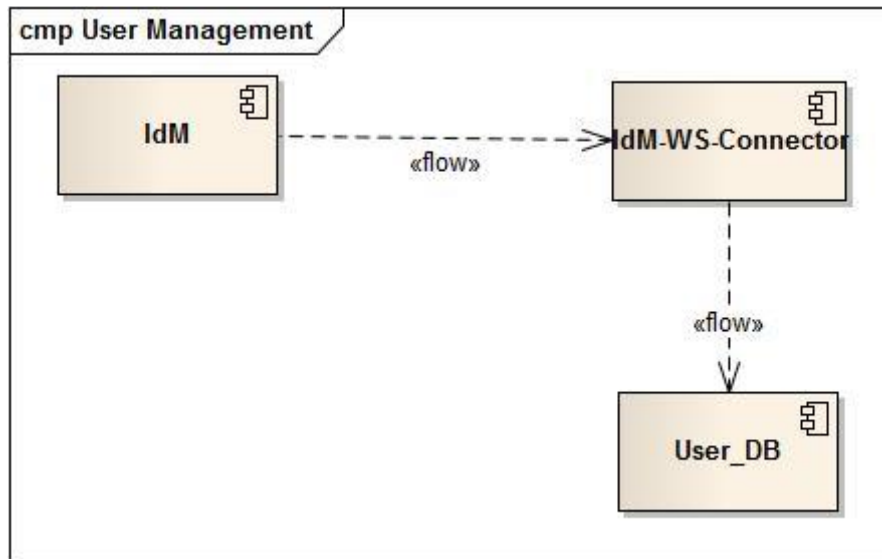


Figure 5-12 - User Management Component Diagram

The Oracle IdM is the identity management tool which is used for provisioning and de-provisioning users to the CSNDC application. It communicates with the IdM-WS-Connector in order to synchronise the CSNDC database with all details about the user. The user management tasks are split between the IdM and the CSNDC application, namely:

- IdM controls the work flow of the user management (creation, editing, disabling, etc.) and in general the access to the portal via an SSO paradigm.
- CSNDC needs to store in its internal database all details of the user that shall be used for two reasons:
 - Implementing RBAC (Role Based Access Control) to the various functionality of the user (i.e. depending on its role, a given user will be authorised/unauthorised to access a given function/sub-function)
 - Storing and managing user detailed attributes that are application specific (e.g. e-mail to be used for the alert, relationships with coastal states or other organisations, etc.)

The integration between the IdM and the User_DB of CSNDC is implemented via Web Service and follow the specification as from [OIM-CSN].

A dynamic view of the process is reported in the following sequence diagram.

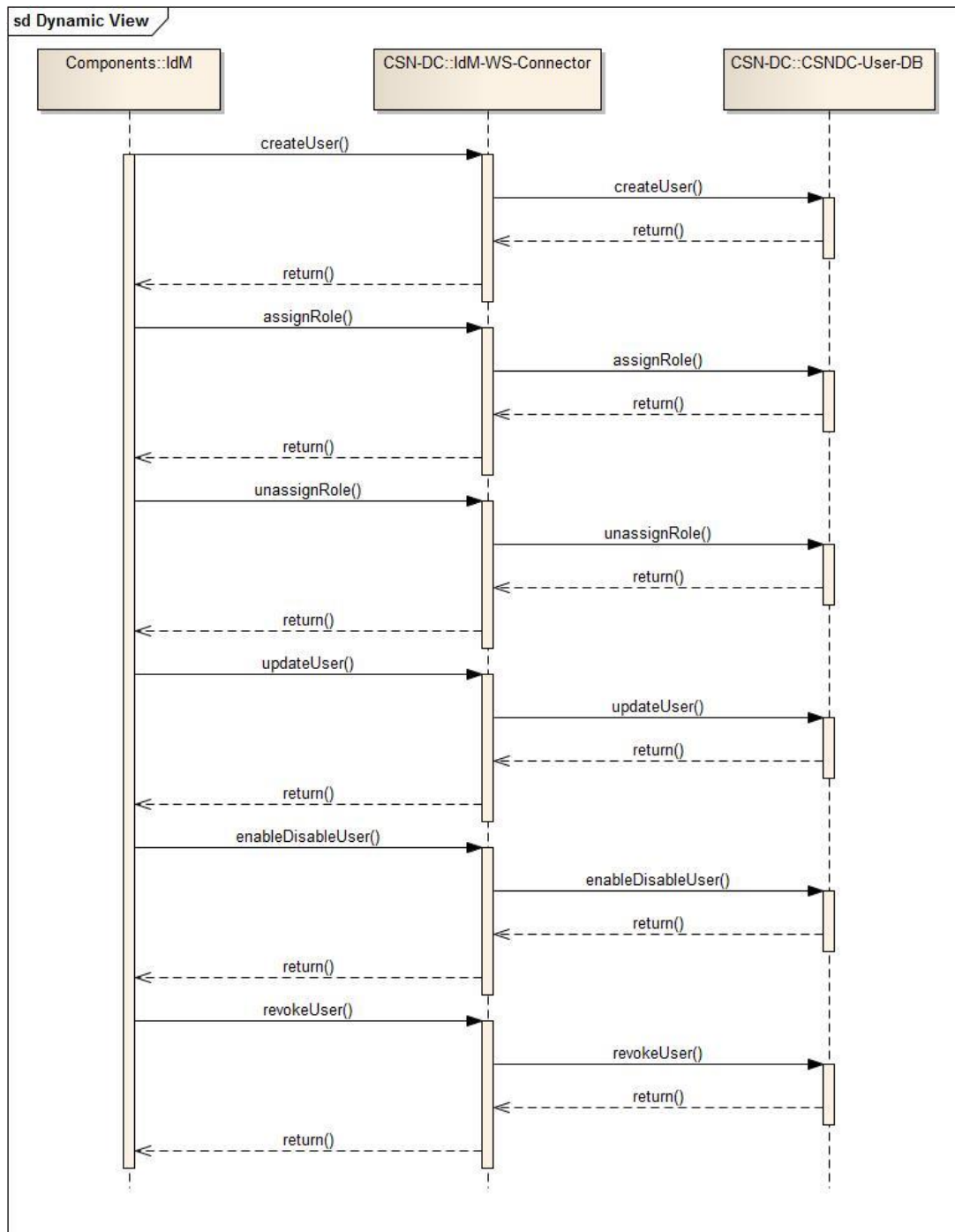


Figure 5-13 - Interaction diagram for the user management

All operations of the web service named **IdM-WS-Connector** are synchronous and return the status for the caller application. The following operations are implemented:

- **createUser**: allows to create a given user. Upon successful creation, the user shall be able to log on the portal. The user details will be registered in the CSNDC database.
- **assignRole**: allows to assign one or multiple roles to a given created user. This is a fundamental step, because a user with no roles will basically have no real access to the application (he/she will be able to log on the portal, but not to the various CSNDC

applications). Once a role has been assigned the user will be able to access the applications, which will be customised according to its profile.

- **unassignRole**: this operation can be used for removing the association between a given user and a role.
- **updateUser**: this operation allows to update a given attribute of the user (e.g. email, contact details, etc.)
- **enableDisableUser**: this operation allows to enable/disable a given user. It is typically used for temporarily un-authorising access to a given application, but it does not remove the user completely from the system, e.g. the user can be re-enabled at a later stage without the need to recreate it.
- **revokeUser**: permanently removes the user.

5.3.1 Technical implementation of the Data Access Policy enforcement

The Data Access Policy will be implemented using a separate component, in order to be as decoupled as possible from the rest of the CSN-DC components. The high level architecture is represented in the following figure.

On the left side there is the *Data Access Component*, which will be developed independently on the existing components (e.g. WUP, POR, etc.). On the right hand side, there is the *CSN-DC Business Component*, generically referring to any of the existing CSN-DC components.

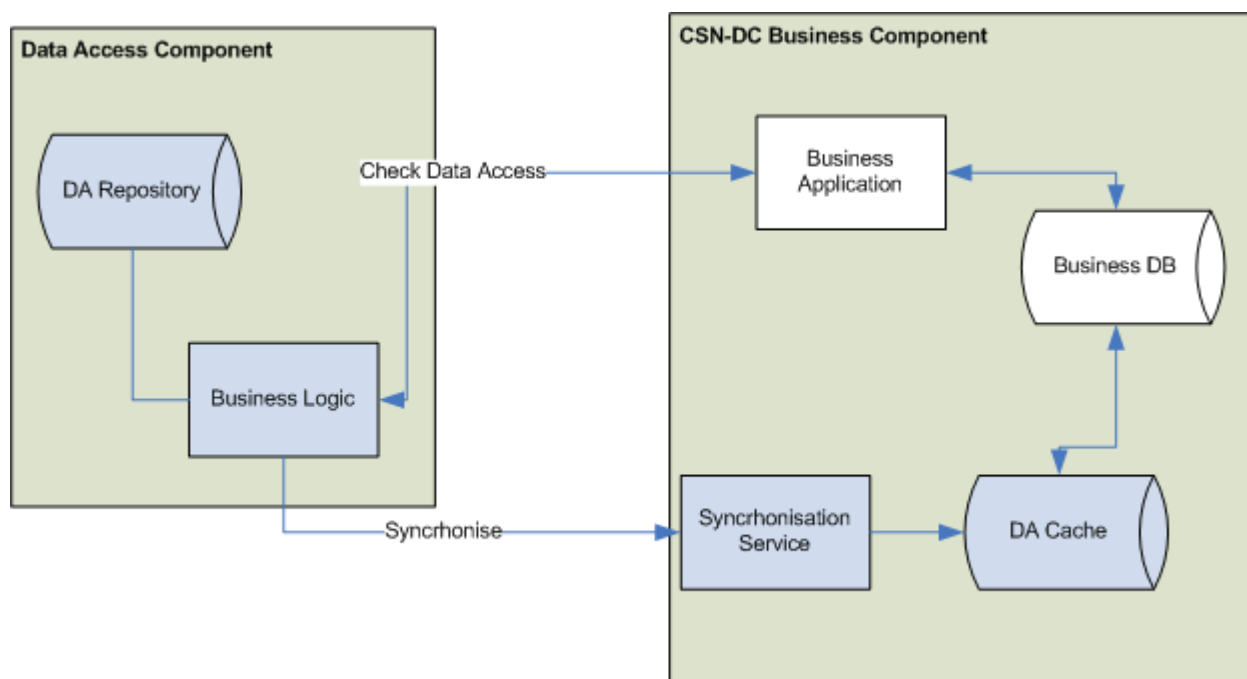


Figure 5-14 High level architecture of the Data Access components

Before accessing the data, the business application shall interrogate the Data Access Component (or the internal DA cache) to perform the filtering / authorisation enforcement.

There are typically 2 use cases in this respect:

- User is querying the data

- User is requesting to perform a specific action (e.g. download high resolution) on a specific data object

The main difference is that the bullet (1) refers to many data, where the bullet (2) refers to one specific data.

For the use case (2) (e.g. requesting to download the data), the Business Application will invoke the Check Data Access service of the Data Access Component. The result of this check could be **true/false**, in case it is false, the action is not authorised (and possibly a popup will notify the user of this). This may happen, for example in case the user has view access rights on some EO data, is generally authorised to perform download (as part of his/her role), but not authorised to download one specific scene. When the user tries to download that scene, an error message is returned.

This paradigm is not recommended in the use case (1), i.e. user querying the data, for 2 reasons:

- *Performances*: the system should first query the database and then, looping for each individual data object, checking the view access rights
- *The query limit effect*: if there is a limit in the number of records that can be returned by the query (always a best practise to avoid overloading by mistake the user with thousands of records), in some circumstances the system may return the maximum number of records (e.g. limit set to 250), but after the check for permission, it may happen that only for few of them the user has visibility right.

Therefore for the use cases related to queries, it is better to filter directly when querying the data, e.g. with a *join* on another table which defines the data access rights for each data granule (e.g. for each service).

In order to de-couple the implementation from the existing components it is proposed to:

- Standardise the structure of the DA Repository
- Keep the DA Repository in an external component and implement a synchronisation with the DA Cache

The DA cache has the same information of the DA Repository, but for performances reasons it is cached in the Business Components. The synchronisation service will be responsible to keep the cache aligned with the DA repository.

Please note that this paradigm is exactly the same as what is currently implemented for the integration between the OIM and the CSN-DC User Database, described in the previous paragraph. In fact the OIM is the official repository of the user details which are synchronised with the CSN-DC database via the provisioning web services, in order to avoid the application to call frequently the OIM.

The logical structure of the DA Repository will be very simple. As explained in the functional description (previous chapter), it will include a number of associations between: users->actions->data.

The logic by which this DA repository is filled in, and its specific implementation, are completely independent on the CSN-DC Business Component and could in principle in the future be replaced by any other implementation, without any need to change the CSN-DC Business Component counterpart, so long as the following 2 interfaces are respected:

- A service (Check Data Access in the figure) which
 - accepts in input:

- identifier of a data object
 - identifier of a user
 - identifier of an action
 - returns in output:
 - true / false
- A service (Synchronise) which replies to a synchronisation request, by providing the updates of the DA Repository for re-synching the DA Cache

5.3.2 Technical Implementation of Operation Based access control

The logical description of this function is reported in § 4.4.4. Here the technical implementation details are reported.

The subset of the data model used for defining the relationship between users and operations is reported below (please note: only relevant fields of SIB_USERS table are reported in this diagram).

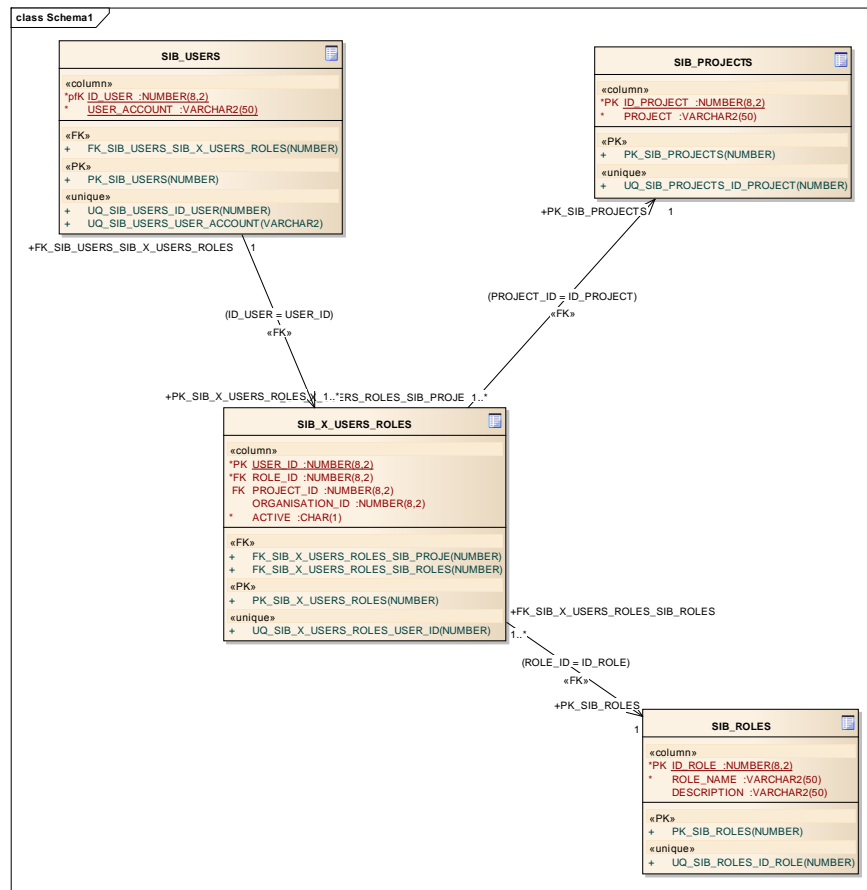


Figure 5-15 – Simplified user model focussed on the user-operations relationships

To say in simple words, a given user can belong to multiple operations (aka projects, in the technical implementation *projects* and *operations* are synonyms) and for each operation the user may have multiple roles. The roles with which a user is associated to a given operations is completely independent on the roles with which the user is associated to other operations.

The association of users to operations can be modified at any time using a dedicated GUI. In order for a user to be able to log in the system, the user shall be associated to at least 1 operation with at least 1 role.

The data model for storing the relationships between service IDs and operations is reported in the following diagram. It is important to stress the following:

- Relation between service ID and primary operation is done at cart level, thus through the table ORDER_MASTERS and the relation is 1 to 1
- Once defined the primary operation association is automatically propagated as a secondary operation association to all individual services in a cart and used as a default unless the secondary operation association is changed individually for one or many services individually
- Relation between secondary operations and service IDs and determined through the ORDER_DETAILS table

Therefore when an order is set into start tasking via the POR, the following happens:

- The user is prompted to provide an Operation to which to assign the whole cart (list of scenes to be ordered)
- This operations is defined at ORDER_MASTER level and is also propagated for all scenes at ORDER_DETAILS level
- When the tasking has started, before the actual tasking starts (before any user has started confirming/allocating the scenes) it is possible to change the secondary Operations, defining possibly a link from a scene to multiple operations. This will override the secondary operation association
- The primary operation association is used for defining the budget centre to be used in the JOU (and is therefore propagated to the JOU via the JMS queues)
- The secondary operation is used for defining access rights
- Association between a single scene and the secondary operation(s) cannot be changed during the tasking and until the scene is successfully delivered.
- After the scene is delivered it can be changed, so that access rights can be granted to users belonging to different operations

The effect of associating scenes to operations is that only users belonging to these operations will be able to access the corresponding data. This includes, all functionalities involving the individual scenes, i.e.:

- Planning: during planning only entitled users are able to access (and therefore allocate, etc.) a given scene, depending on the operations setting
- GIS Viewer: all scene based functions (search, download, etc.) will be only accessible to users on the basis of the association to operations

The EMSA users with roles UP02 and UP21 are considered as super users and therefore have access to all data regardless of the operations setting.

In addition the setting of operations for a given scene has been also propagated into the OGC CSW response so that it is clear to systems accessing the data which are the operations for which the data were ordered. This would allow, for example to implemented on the client calling applications a possible security strategy.

An example of CSW response including the operations association is reported below.

```
<csw:GetRecordsResponse version="2.0.2">
<csw:RequestId>11</csw:RequestId>
```

```
<csw:SearchStatus timestamp="2014-06-18T10:07:25Z"/>
<csw:SearchResults elementSet="full" expires="2014-06-18T10:07:25Z" nextRecord="0" numberOfRecordsMatched="1"
numberOfRecordsReturned="1" recordSchema="http://twls10:7021/vcat-
csw/services?service=CSW&version=2.0.2&request=DescribeRecord&typeName=%7Burn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0%7D:ExtrinsicObject">
<rim:ExtrinsicObject
id="urn:eop:CSNDC:product:RS2_20110330_162728_0045_SCNA_HH_SGF_126262_3562_4984131.zip_0001"
isOpaque="false"
lid="urn:eop:CSNDC:product:RS2_20110330_162728_0045_SCNA_HH_SGF_126262_3562_4984131.zip_0001"
mimeType="application/octet-stream" objectType="urn:x-ogc:specification:csw-ebRim:ObjectType:EO:EOPProduct"
status="valid">
<rim:Slot name="urn:ogc:def:ebRIM-Slot:OGC-06-131:status"
slotType="string"><rim:ValueList><rim:Value>ARCHIVED</rim:Value></rim:ValueList></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:multiExtentOf" slotType="geometry"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:centerOf" slotType="geometry"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:acquisitionStation" slotType="string"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:acquisitionDate" slotType="dateTime"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:orbitNumber" slotType="int"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:productType" slotType="string"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:orbitDirection" slotType="string"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:acquisitionType" slotType="string"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:acquisitionSubType" slotType="string"></rim:Slot><rim:Slot
name="urn:ogc:def:ebRIM-Slot:OGC-06-131:doi" slotType="string"></rim:Slot><rim:Slot name="urn:ogc:def:ebRIM-
Slot:OGC-06-131:beginPosition" slotType="dateTime"></rim:Slot><rim:Slot name="urn:ogc:def:ebRIM-Slot:OGC-06-
131:endPosition" slotType="dateTime"></rim:Slot>
<rim:Slot name="urn:CSNDC:CSNDC-Slot:operations" slotType="string">
<rim:ValueList>
<rim:Value>CleanSeaNet</rim:Value>
<rim:Value>PlatFormMon</rim:Value>
<rim:Value>test-operation</rim:Value>
<rim:Value>test-operation-2</rim:Value>
</rim:ValueList>
</rim:Slot>
<rim:ContentVersionInfo versionName="1.1"/></rim:ExtrinsicObject></csw:SearchResults></csw:GetRecordsResponse>
```


5.4 PMA - Process Management

The most interesting dynamic analysis of the PMA are related to the systematic processing. This can be illustrated starting from the diagram of Figure 5-1. Whenever a data of any type is ingested into the DAM, a notification is sent to the PMA and the PDE, that have their internal knowledge about possible tasks to be executed on the basis of the arrival of a certain type of data.

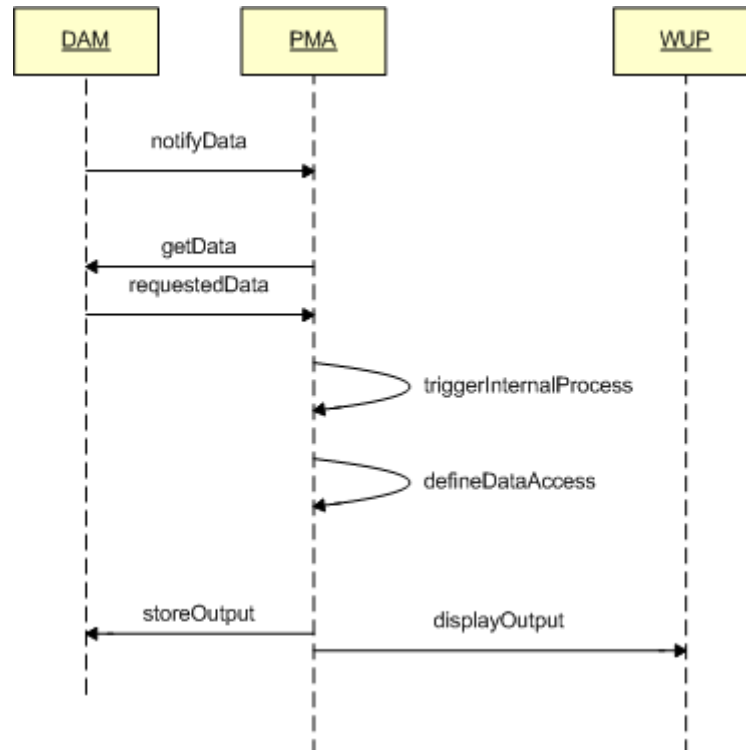


Figure 5-17 - Ingestion processing and display of Service Provider data

Therefore the typical sequence, starting from the point where the data has been successfully ingested into the DAM :

1. a notification message is sent to the PMA
2. According to a pre-configured workflow, the PMA does the following:
 - a. gets the data from the DAM
 - b. triggers the execution of an internal process
 - c. defines the data access rights
 - d. stores the output into the DAM
 - e. sends the data to the WUP for synchronous display

In fact the PMA allows to define complex workflows, that combine any sequence of internal processes/services, externally hosted processes/services and ingestion processes.

Let's imagine for example that the system shall do the following type of processing:

1. ingest some SAR data
2. trigger oil spill detection internal process
3. on the basis of the oil spill detected features execute an externally hosted oil spill model
4. retrieve the results of the oil spill model and:

- define the data access rights
- store the output into the DAM for successive further consultation
- display the output in the WUP

This scenario is covered by the following sequence diagram.

In this scenario, the External Hosted Process is an oil spill model, which is made available outside the CSN-DC, but can be executed as a web service (e.g. via a SOAP interface).

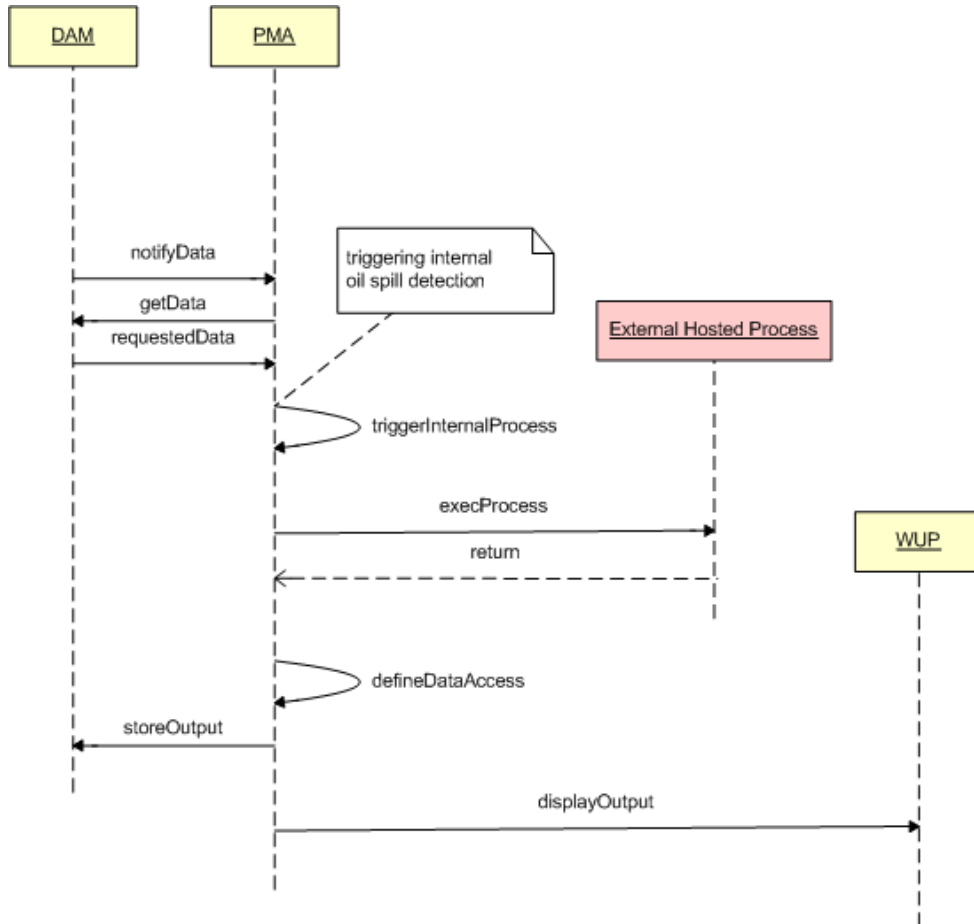


Figure 5-18 - Complex chain including execution of an external hosted process

As described in section 4.4.3, the Thin Layer implementation of the PMA, due to its characteristics of efficiency and robustness will be used for implementing the systematic processing, where availability and timeliness are a critical factors. On the other hand, the KEO implementation will be used for defining ad-hoc manual processing, where new processing chains can be experimented and modified easily.

The next scenario illustrates a situation slightly more complicated. In this sequence of operations the PMA receives some data (e.g. SAR images and oil spill) and request some STIRES data to be used for complementing the analysis of the SAR data and associated oil spills.

The following will happen:

1. DAM notifies PMA that some data has been successfully ingested
2. On the basis of its internal configuration, the PMA gets some data from the DAM and starts the analysis (let's assume that SAR data and associated oil spills have been received from the SPs)

3. The PMA extracts some info from the data package, e.g. the geographical boundaries of the SAR images over which to query the STIRES data and calls the IIF for retrieving the STIRES data
4. The IIF interfaces with the STIRES and retrieves the necessary AIS data
5. The AIS data from STIRES are inventoried into the DAM, which in turn notifies the PMA
6. The PMA can couple the STIRES data with the already available SAR and oil spill data in order to perform a complex analysis of the data. The rest of the process is similar to the other diagrams above (store results, define data access policies, display on the WUP, etc.)

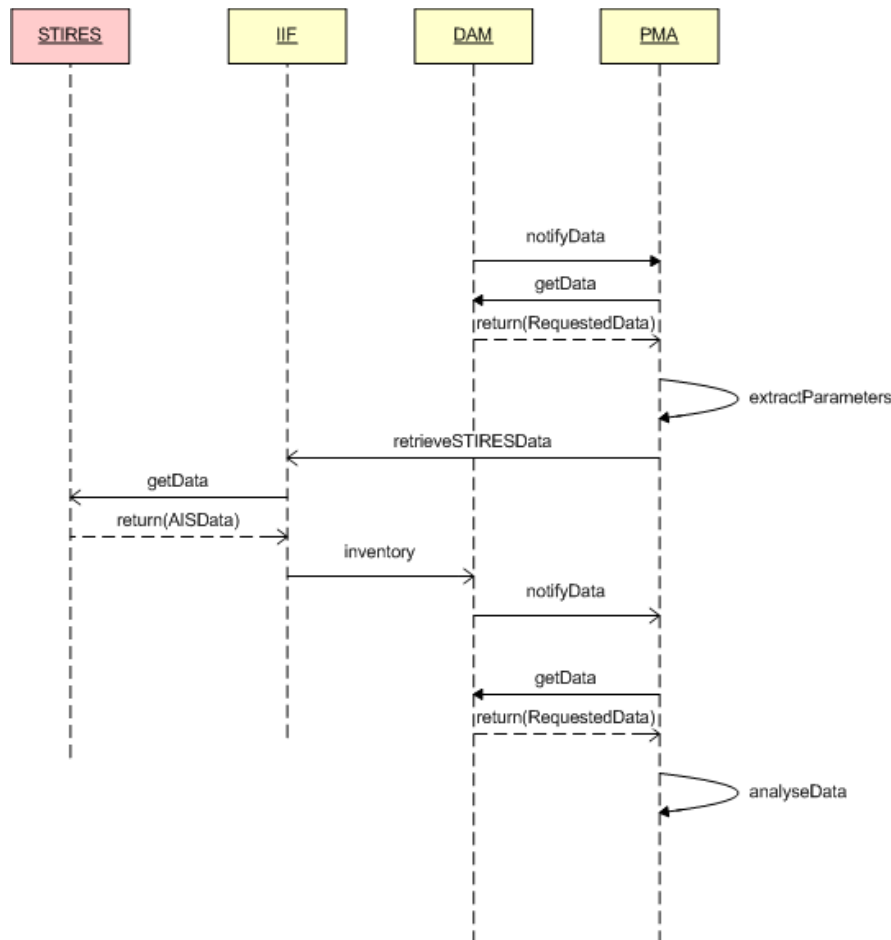


Figure 5-19 PMA processing and request for STIRES data

This implementation is preferred to a simpler scenario in which the PMA directly calls the STIRES for extracting the relevant parameters, because it allows for performing the complete data ingestion and storage lifecycle for the STIRES data (they will be ingested into the DAM and made available for any further display or analysis).

5.5 WUP - WEB USER PORTAL

The dynamic analysis of the WUP is reported in § 4.6 together with the logical implementation view, as it integrates the general understanding of the WUP behaviour. Therefore it will not be reported here.

5.6 PDE - Product Delivery

5.6.1 Component diagram and main interfaces with other components

Alerting component diagram is depicted below.

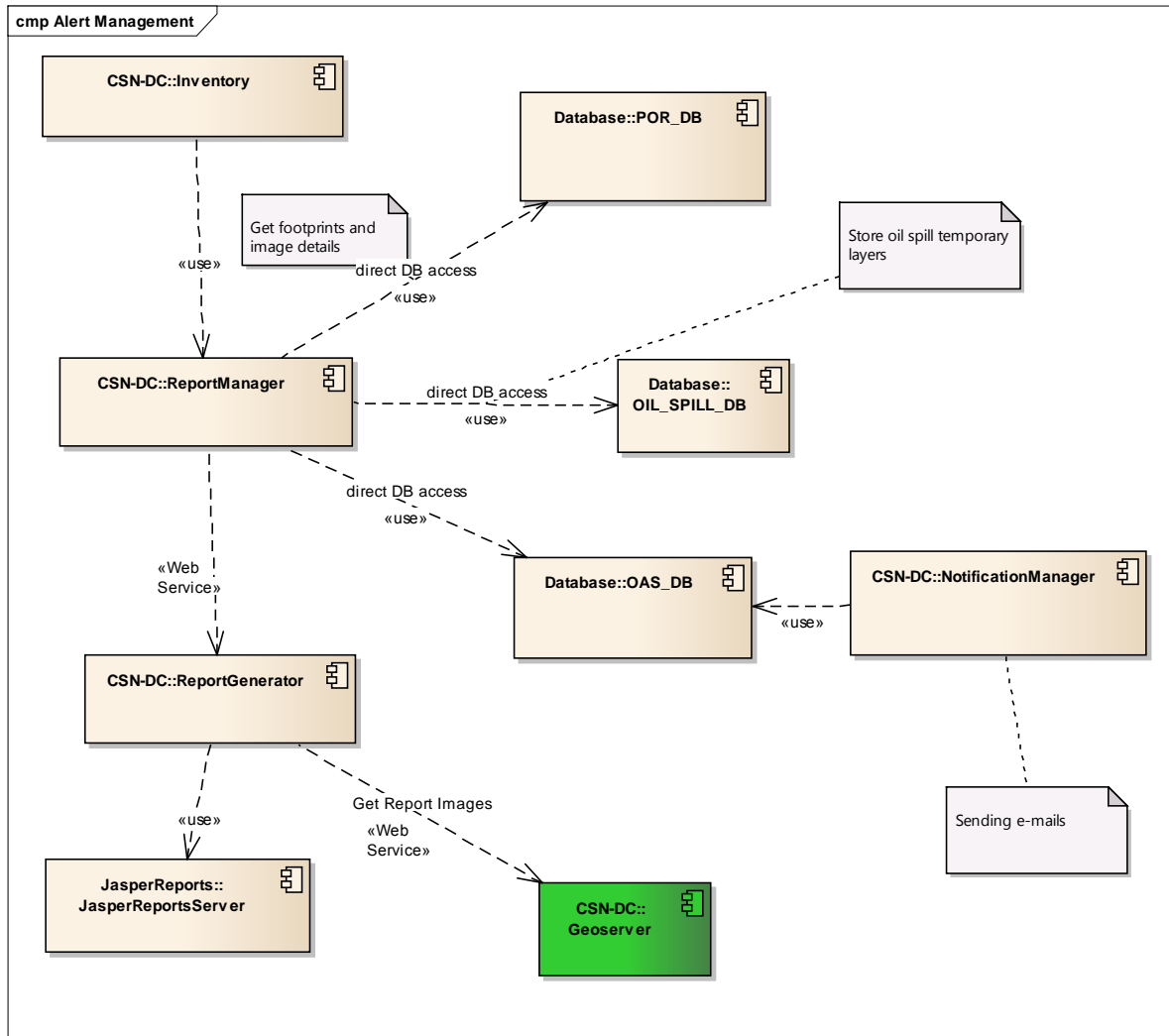


Figure 5-20 – Component diagram for Alerting and its main interfaces

The following interfaces are described:

1. Inventory->Report Manager: the inventory triggers the report manager upon reception of OSN data, this is a simple system call
2. ReportManager->POR_DB: report manager accesses the POR_DB to retrieve info about the corresponding scene, for example, the footprint and other details.
3. ReportManager->OIL_SPILL_DB: report manager accesses the OIL_SPILL_DB mainly to store temporary oil spill vector layers which will be later exploited for drawing the images for the PDF report.

4. ReportManager->OAS_DB: report manager accesses the OAS_DB to get info about alerting areas, and other details and to finally store the results of the alert generation process (e.g. the PDF report, the list of recipients, etc.)
5. NotificationManager->OAS_DB: the notification manager gets from the OAS_DB the information necessary to send e-mail (attachment PDF, subject and body recipients, etc.).
6. ReportManager->ReportGenerator: this is a Web Service request which will return to the caller the exit status of the report generation procedure and the result itself.
7. ReportGenerator->Geoserver: the report generator uses geoserver to create the various imagenttes that are attached to the report (all expect the clip image which comes already as a PNG in the payload). Geoserver is preconfigured with a number of ancillary layers that are filled in with temporary data (oil spills, detected vessels, alerting areas) by the Report Manager (which is also performing the clean up of such data upon successful report generation). The ReportMANager prepares a URL which embeds the geoserver request (basically a getMap with various layers in overlay) necessary to create the imagenttes to be added in the report
8. ReportGenerator->JasperReportServer: this is an intra-component I/F, the report generator embeds the necessary libraries from the Jasper COTS and uses them.

5.6.2 Dynamic behaviour

The following sections report some analysis of the dynamic model of the PDE. In the following text the term *product* will be used to refer to all the data items that can be requested by the user for systematic and/or ad-hoc delivery.

One of the major data flows is the one dealing with a data set service subscription.

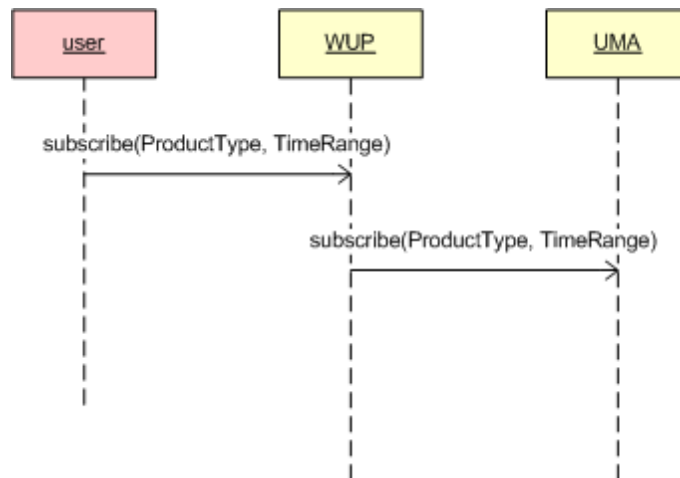


Figure 5-21 - Management of Subscriptions

The followings steps are executed:

1. a User subscribe to a service data set via WUP: the user may specify the following parameters:
 - a. product type

- b. time range of subscription (the sensing time range of the data the user subscribes to)
2. WUP stores the subscription details into the UMA, which stores all information about the users

NOTE: physically no PDE component is involved in the above use case scenario, but it has been included here as it is a process that logically is part of the product delivery.

The automatic process for handling the subscription when products are received is illustrated in the following sequence diagram:

1. Whenever a new product is ingested the IIF performs inventory into the DAM
2. The DAM sends a data notification to the PDE
3. The PDE gets the subscription information from the UMA (see previous figure) and checks the subscription
4. If there is at least one user subscribed to that type of data the PDE initiates the delivery, instructing the IIF in order to send the data
5. The IIF distributes the data to the user
6. Asynchronously, on end of delivery of data to the user the IIF notifies the PDE, also indicating the status (successful/unsuccessful)
7. The PDE logs the data delivery outcome in the JOU (e.g. for billing purposes, etc.)

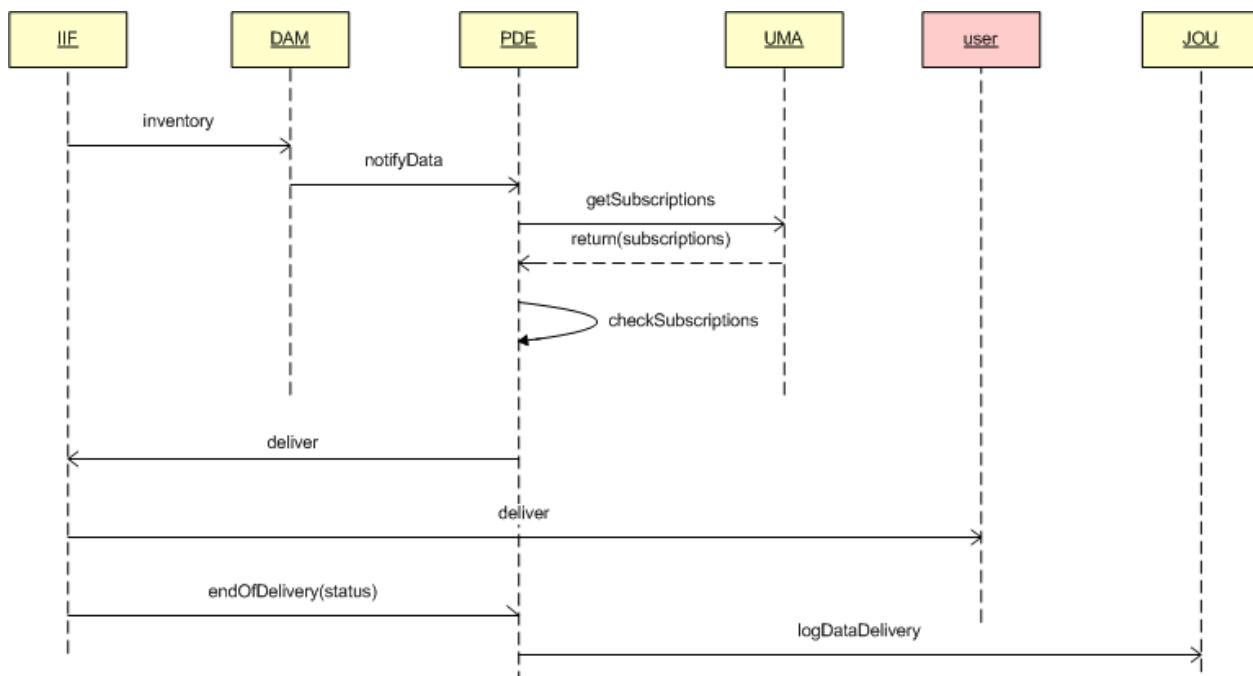
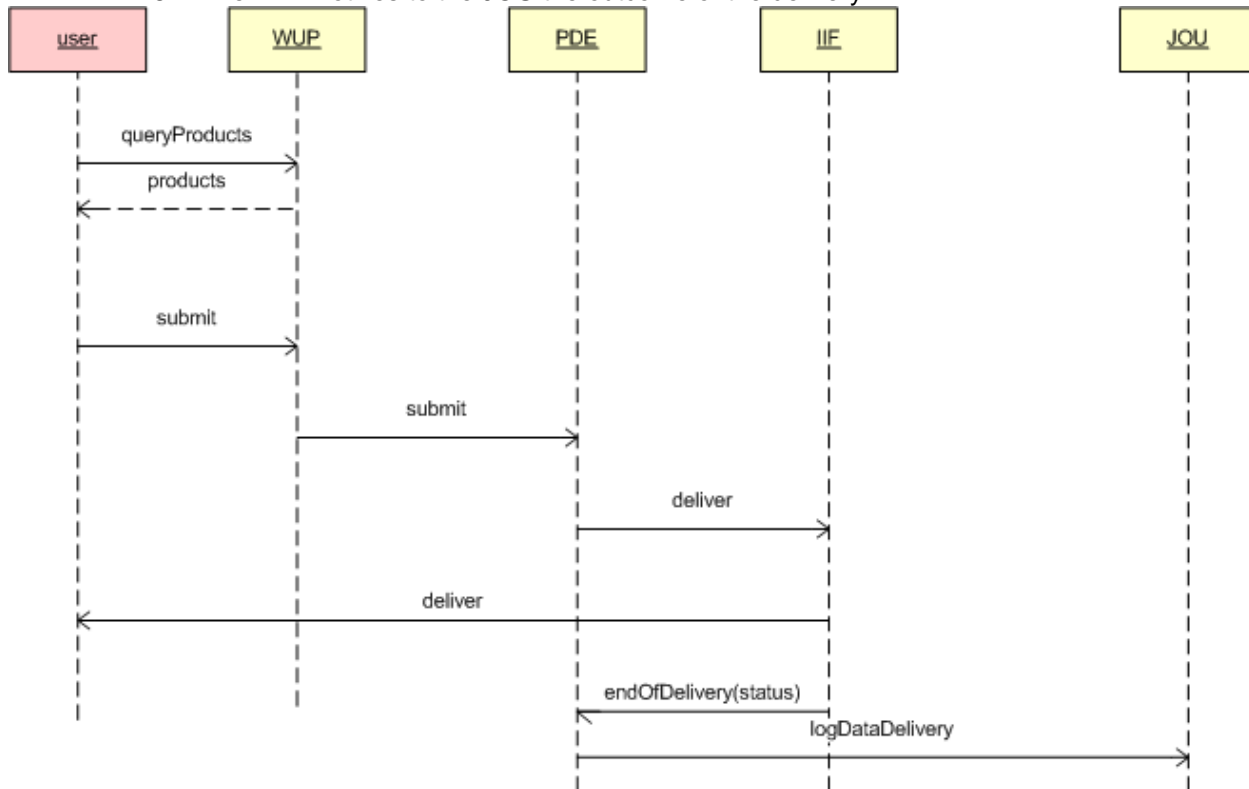


Figure 5-22 Systematic distribution of data

The interactive data delivery is performed according to the following figure.

1. The user queries the available products on the WUP (using all WUP query parameters available, e.g. region of interest, product type, time range, etc.)
2. The WUP returns and displays the products according to the query performed
3. The user submits a request for a certain (1 or many) product(s)
4. The request is submitted to the PDE, which manages the orders

5. The PDE sends a request for delivery of the requested data to the IIF; from this step onward the mechanism is the same as in the figure above, i.e.:
6. The IIF delivers the requested data
7. Asynchronously the IIF notifies the PDE upon end of delivery
8. The PDE notifies to the JOU the outcome of the delivery



The following diagram depicts the sequence of steps for the generation of a report. This represents the case of a report-alert automatically generated as part of some systematic processing activity:

1. The IIF initiates the report generation, passing all information about the scenario (e.g. oil spill detected/clean sea/vessel detection, oil spill metadata and characteristics, etc.)
2. The PDE retrieves the user configuration from the UMA which includes:
 - a. conditions for generating a report, e.g. size of the oil spill, confidence level, distance from the coast, etc.)
 - b. type of report (e.g. short report, long report, report communication channel, etc.)
3. the PDE retrieves the report data from the DAM. The details of this operation, which in this diagram is simply indicated with the *retrieveReportData* message have already been illustrated in Figure 5-3 (interaction between the DAM and the PDE)
4. based on the configuration information the PDE produces a number of reports (one for each type of different configuration of reports)
5. the reports are created according to the user configuration and dispatched to the various coastal state users.
6. The PDE stores the dispatched reports into the DAM
7. The reports sent are logged into the JOU

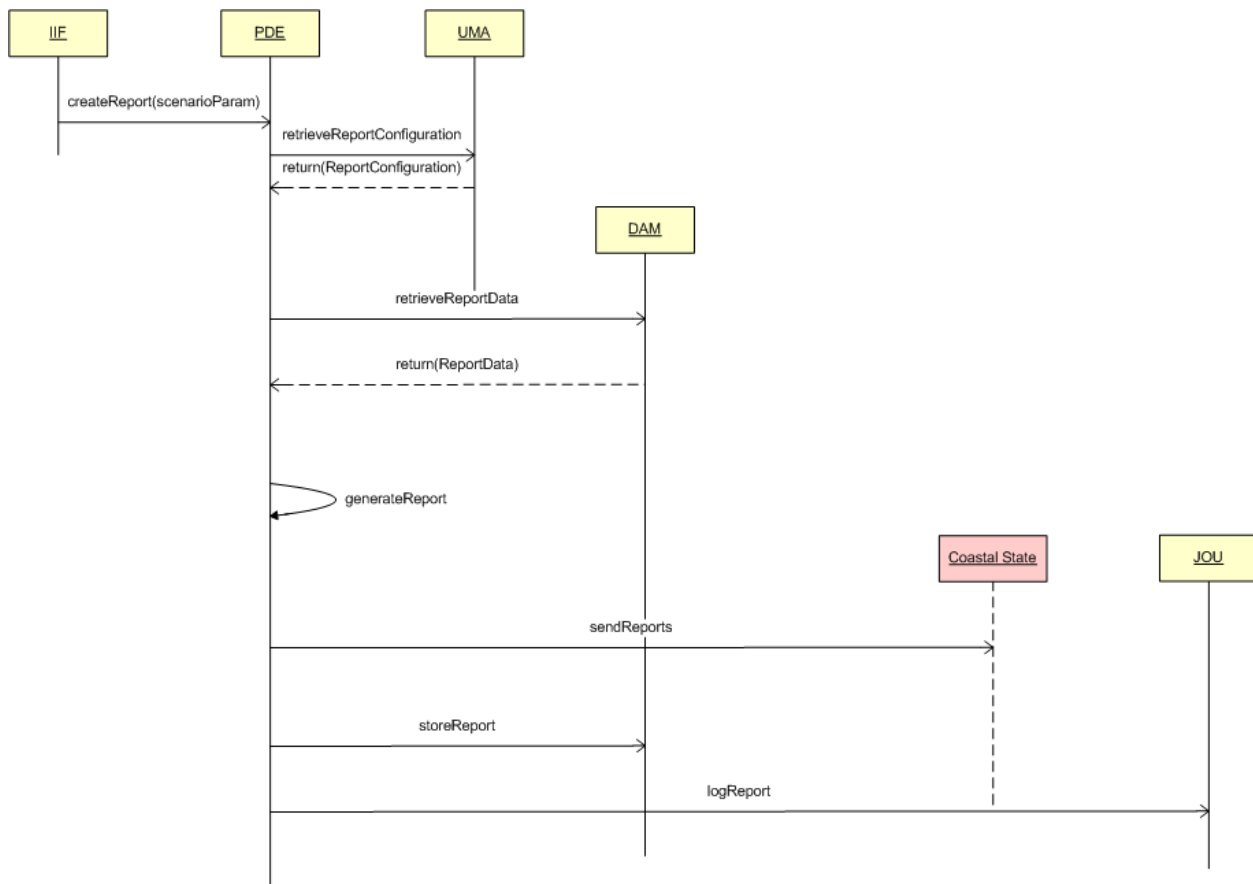


Figure 5-23 Generation of reports and alerts

5.7 POR Planning and ordering

5.7.1 Component diagram and main interfaces with other components

The component diagram is reported below.

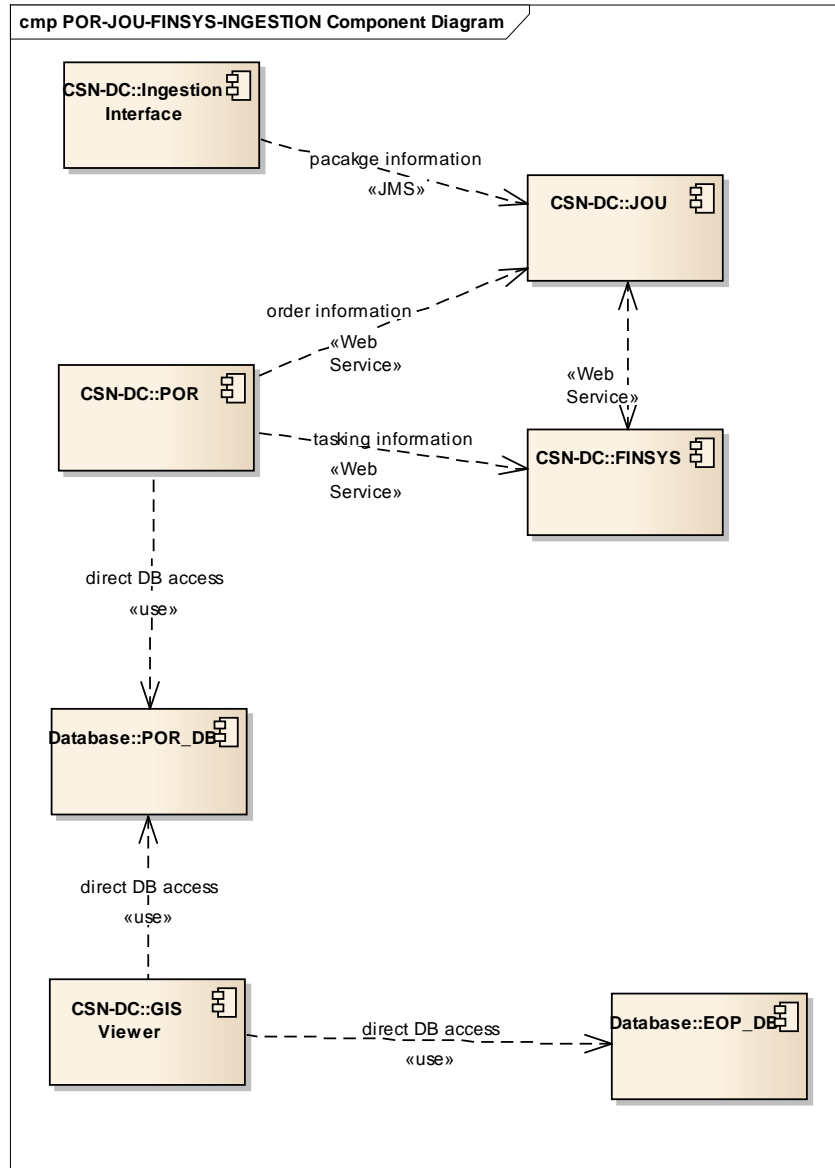


Figure 5-24 POR JOU FINSYS and GIS Viewer component diagram

The main flows are described hereafter:

- Ingestion interface -> JOU: interface based on JMS messages, describing the content of the data package received, timeliness, etc. Messages are sent to a JMS queue on WL.
- POR -> JOU: various messages are sent to the JOU during the planning and ordering process. The interface can be either JMS or Web Service (configurable). Currently JMS is configured.

- POR -> FINSYS: various messages are sent to the JOU during the planning and ordering process. The interface can be either JMS or Web Service (configurable). Currently Web Service is configured.
- JOU->FINSYS: JOU and FINSYS exchange data for business workflow using web services.
- POR->POR_DB: this is an interface between 2 elements of the same component, in the sense that the POR_DB is the persistence of the data used by the POR. As such this interface is an intra-component interface rather than an inter-component one.
- GIS Viewer->POR_DB: the GIS Viewer accesses the POR database to retrieve some information, mainly regarding the status of the ordered data. This interface consists in directly accessing the database.
- GIS Viewer-> EOP_DB: this interface is used for retrieving information about the EOP data, for the complex business logic of the GIS viewer. The EOP_DB basically implements a CSW ebRIM EO catalogue and is exposed as such to external users, but for performance and flexibility reasons (need to perform complex joins between this database and various other tables) a direct database access is preferred

5.7.2 Dynamic behaviour

The sequence of actions underlying the process of planning and ordering is exemplified in the following sequence diagrams. The first diagram shows how the default coastal state requirements can be set using the POR. The Coastal state sends the requirements to the CSN-DC Service Desk (CSN-SD), e.g. sending some shapefiles with the areas to be covered and indication of the coverage density.

The CSN-DC validates the requirements, by analysing them and checking that no error has been made and loads the default requirements into the POR.

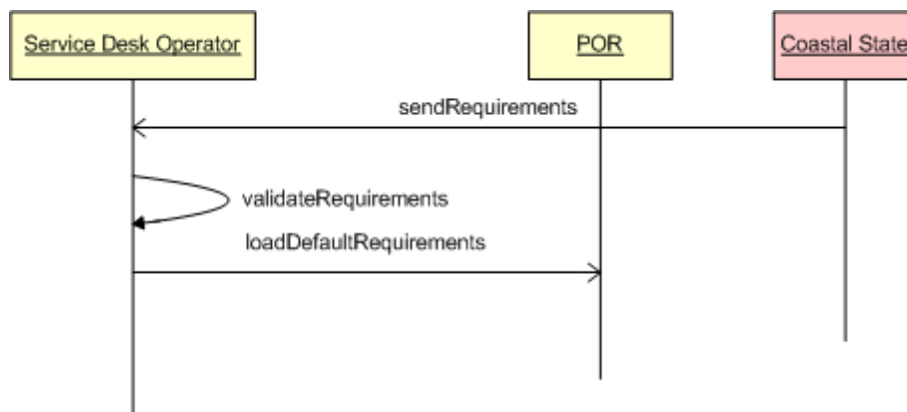


Figure 5-25 – Setting the default coastal state requirements using the POR

The next diagram (Figure 5-26) depicts the operations by which the CSN-SD defines the list of scenes corresponding to the coverage requirements defined by the Coastal States. The SD displays

the requirements on the POR. The requirements are visualised in terms of areas to be covered and monthly frequency. Optionally, the SD edits the requirements (for example upon specific request of a Coastal State). The SD retrieves the query files (these are the query files in the native format supported by various tools, e.g. the EOIL query files) from the POR, generated using the coverage requirements. These query files are in the format suitable for being imported into the mission planning tools to be used for starting the initial planning of the scenes (e.g. EOIL query format). After loading the files into the mission planning tool, the SD performs the query and obtains the list of scenes to be used for matching the requirements. This list can be exported into a format which is suitable for the POR and successively loaded into the POR for further analysis.

The follow on depends on whether the order covers ESA data or other data. The case of ordering ESA data is covered by the Figure 5-27. The SD operator retrieves the order in the format of an EOIL shopping cart format and loads it into the EOIL tool. Successively the EOIL tool is used for actually submitting the data order to ESA.

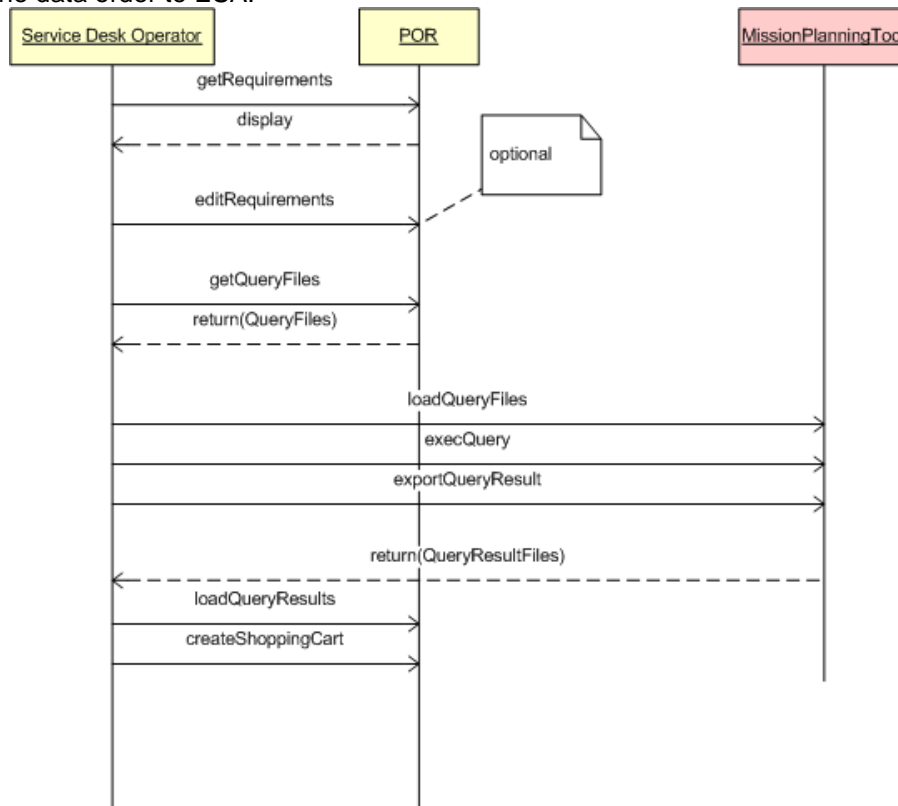


Figure 5-26 – CSN-SD creates the list of scenes allocated by Coastal States in the various formats.

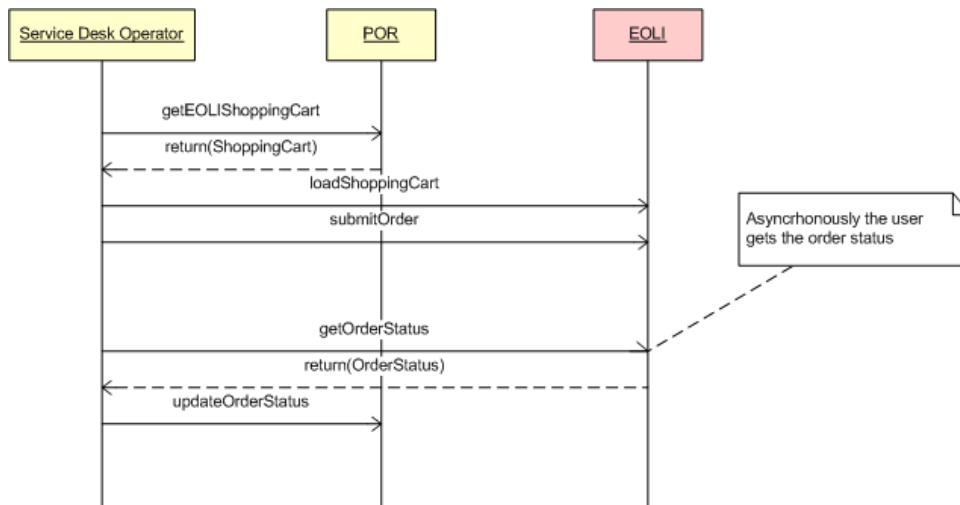


Figure 5-27 – CSN-SD sends EOLI order and update order status

The case of an order to other providers is covered by the Figure 5-28. In this case a more complex interaction with the satellite operator takes place:

- The POR automatically notifies the satellite operator that some data have been selected for order
- The satellite operator logs onto the POR and exports the shopping cart
- The satellite operator updates the status of the individual scenes on the POR (changing the status flag)

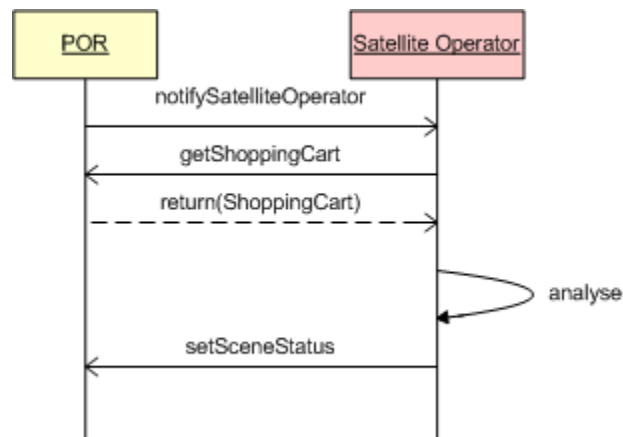


Figure 5-28 POR ordering data (tools different from EOLI)

The last sequence diagram of the POR illustrates how the POR can actually finalise the order. This last procedure involves various actors: the SD, the satellite operators, the service providers, the coastal states. The following actions are taken:

- Coastal states are notified of the list of scenes predefined and may choose to edit scene status (e.g. allocate or reject the scenes)
- The allocated scenes are displayed on the POR by the CSN-SD
- The tasked scenes are notified both to the satellite operators and to the service providers

- Satellite operators and service providers asynchronously may confirm or reject the tasked scenes, by setting a confirmation status. There is not predefined order in which they send this confirmation, but in any case confirmation of both parties is mandatory for carrying on the task. In particular for the service provides a cascading approach is carried out, whereby there is for each scene a default service providers and, possibly, some additional service provider who may confirm acquisition of a certain scene overpass. This means that if the default SP rejects acquisition of a certain scene, the other additional service providers may confirm the scene acquisition on their place.
- At the end of this process, in case all parties successfully confirm the acquisition of the scenes the scene license orders and the processing orders are sent respectively to the satellite operators and to the service providers
- Finally the order forms and the scene licenses are logged into the JOU, they will be used at later stage for invoicing purposes.
- At a later stage (after ordering/tasking) a Coastal State can request a scene (or a number of scenes) to be de-allocated. This communication should be handled via the COM in a structured way

Please also compare this last step with Figure 5-4, which illustrates how, at the end of the reception of the data packages from the service providers, the operation timeliness is transferred to the JOU, along with the order ID, in order to close the loop about the service provision. The order ID will allow to define the correlation between the orders that have been placed in this context and the actual execution of the services.

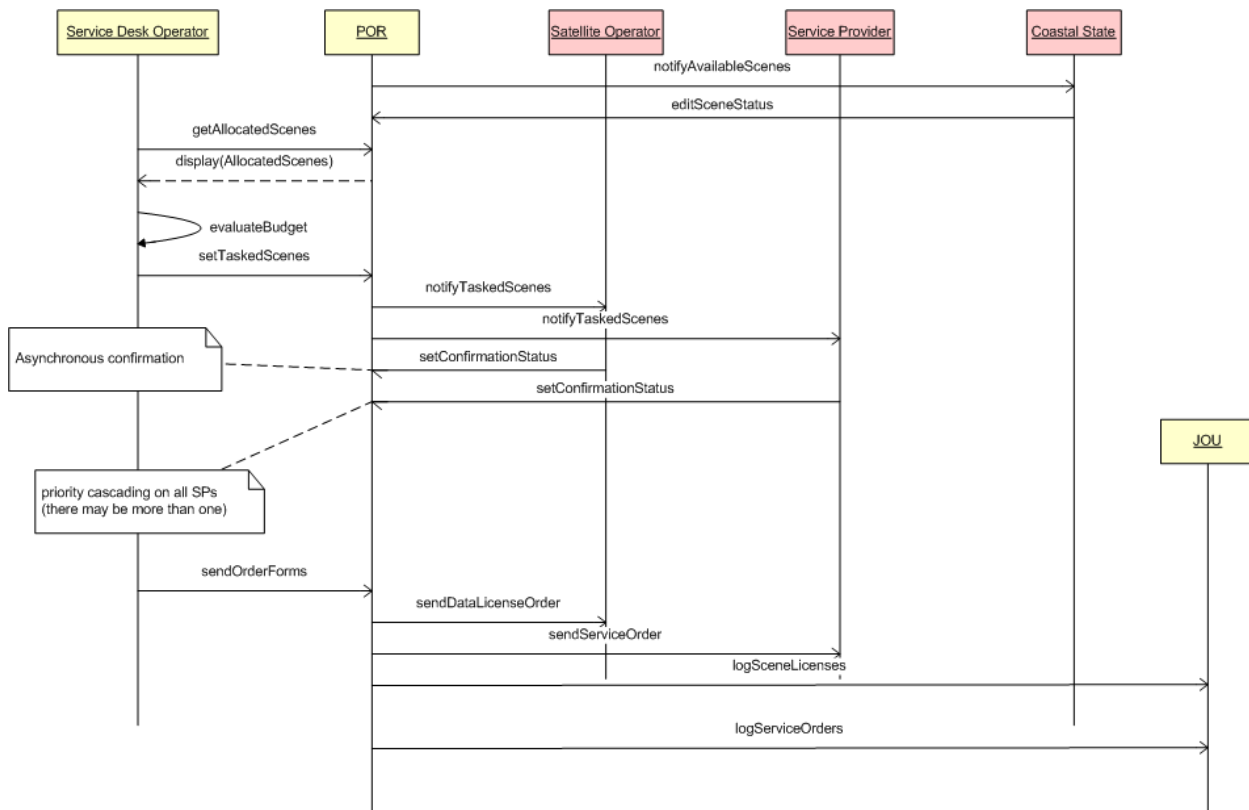


Figure 5-29 Finalisation of an order from the POR

Once an order has been allocated by all stakeholders according to this workflow described above, the last macro-steps consists in the approval of the order by the financial officer and by the authorising officer. The following steps are performed:

1. The CSNDC Project Officer (alias the Service Desk) starts the approval phase
2. The Financial officer accesses the POR and performs the following operations
 - a. View budget
 - b. Approve / Reject the cart
3. Upon Financial Officer approval, the cart is set in status approved by FO and the workflow resumes from step 5
4. Upon Financial Officer rejection, the cart workflow is stopped
5. The Authorising officer accesses the POR and performs the following operations:
 - a. View budget
 - b. Approve / Reject cart
6. Upon approval the cart is finally tasked (continue with step 8)
7. Upon Authorising Officer rejection, the cart workflow is stopped
8. Upon successful approval, the POR sends a message to the FINSYS which, in turn generates and send the tasking lists and tasking forms to the appropriate repositories (see § 4.11for details).

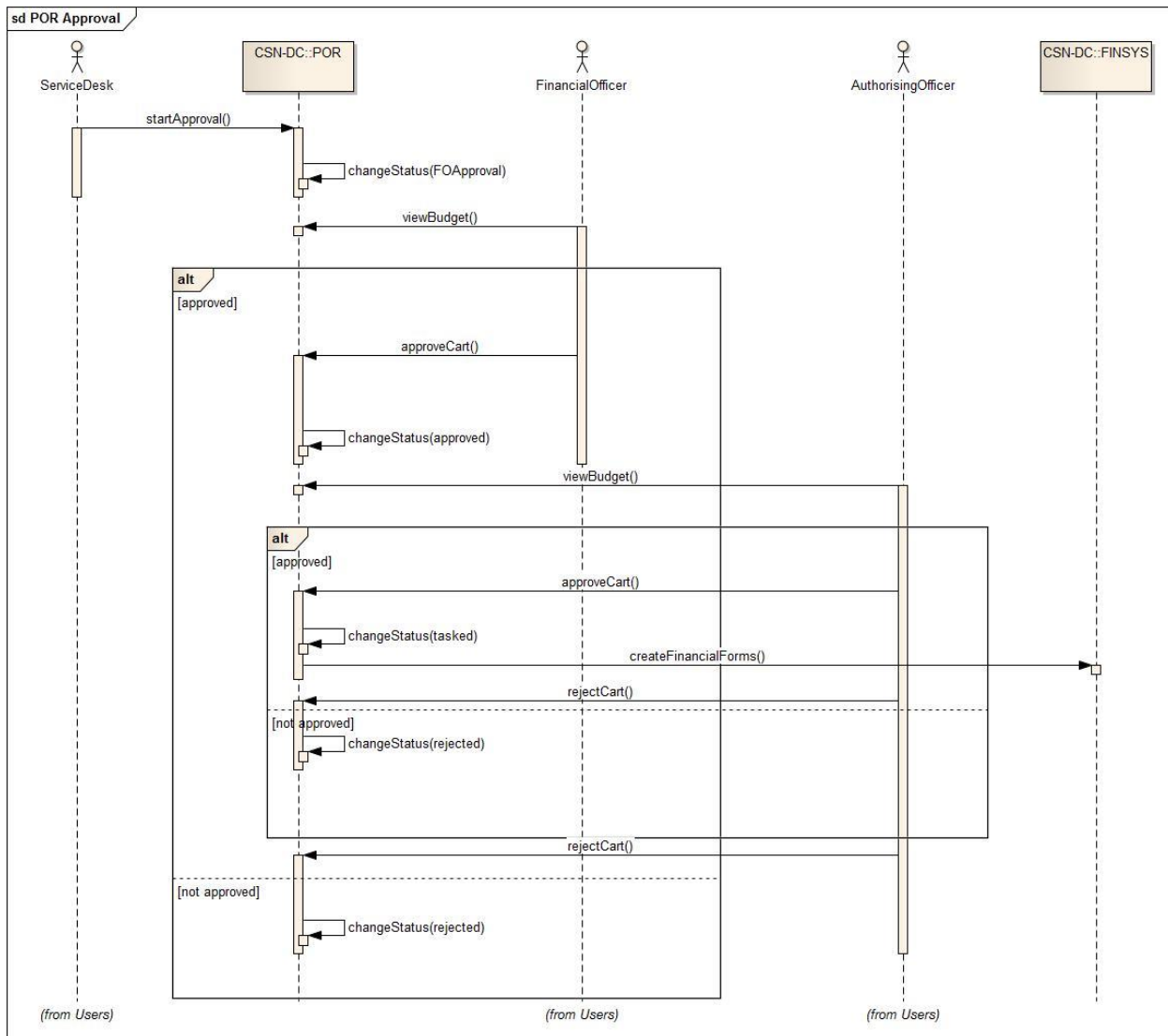


Figure 5-30 - POR approval sequence

5.7.3 Description of the message exchanges between POR and JOU/FinSys

The POR exchanges various messages with the JOU and the FinSys during the complete workflow for ordering and tasking scenes. It is fundamental to understand this workflow, in order to following the messages exchanges and their implications.

During the allocation and tasking process there are a number of messages that are exchanged with the FinSys and JOU in order to correctly synchronise all the information which is necessary for the tasks of the various elements.

The sequence of operations reported below is not the complete sequence of operations actually performed by the POR, but only focusses on those which trigger information exchange with the JOU and FinSys.

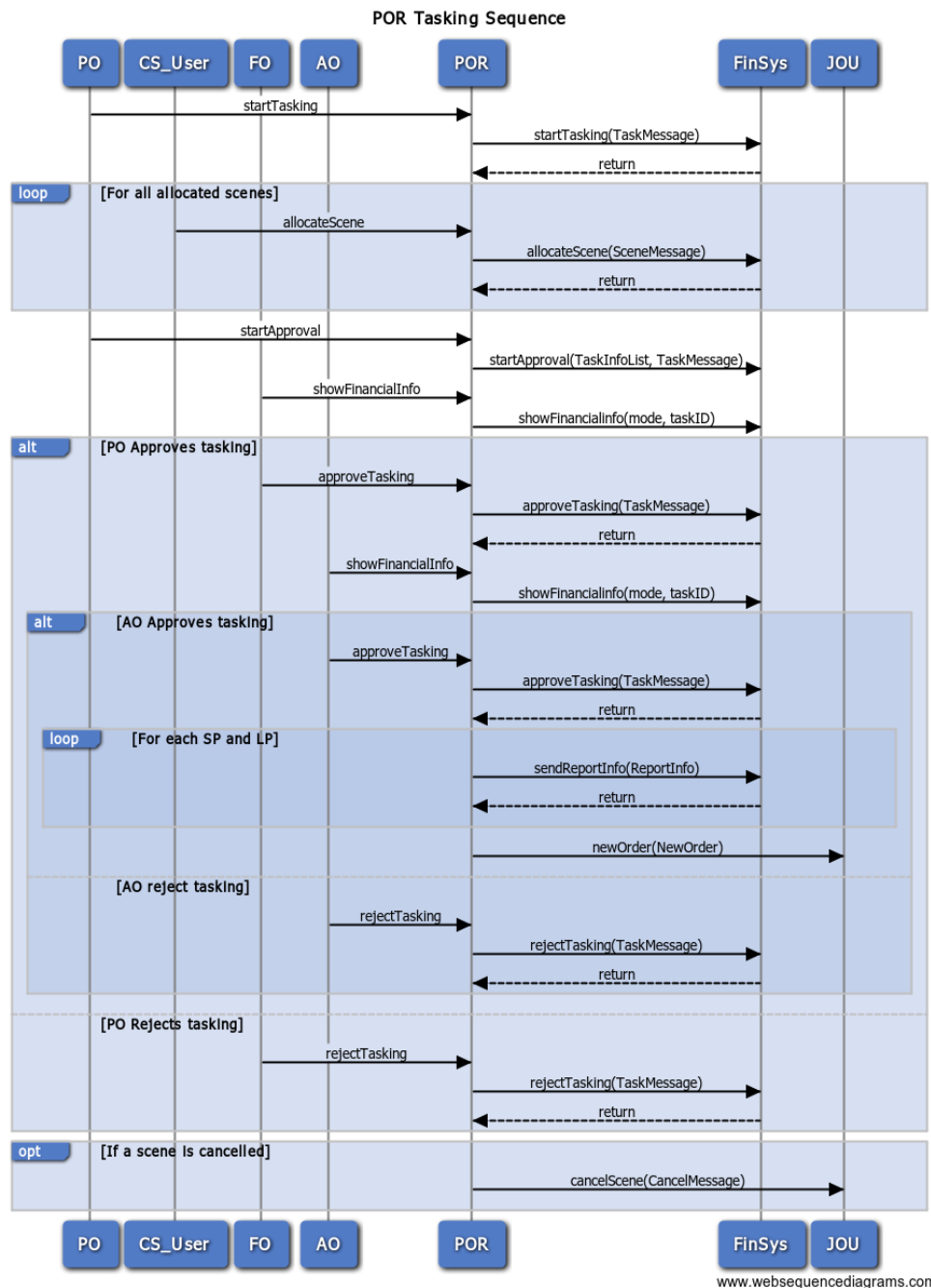


Figure 5-31 Information exchange between POR and JOU/FinSys

The first message that occurs is when the Project Officer (PO) performs the Start Tasking. A message is sent to the FinSys.

The next message is transmitted to the FinSys every time there is a new scene allocated by a coastal state. This happens for all scenes that are allocated by a coastal state (e.g. potentially for all scenes of a given POR cart). The message is of type SceneMessage. The next message is sent when the approval phase is initiated by the PO, with a message

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<TaskMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="POR_JOU_FINSYS_v1.3.xsd">
  <MessageID>2011-03-24T18:21:21+01:00</MessageID>
  <TASK>
    <Identifier>9996</Identifier>
    <TaskFormNumber>1</TaskFormNumber>
    <TaskStatus>Planned</TaskStatus>
    <Description></Description>
    <Scenes>
      <Scene>
        <frameID>9911328</frameID>
        <project>PROJECT_1</project>
        <beginPosition>2012-07-20T20:03:28.380Z</beginPosition>
        <endPosition>2012-07-20T20:03:28.380Z</endPosition>

        <satelliteOperator>ESA</satelliteOperator>
        <serviceProvider>Ksat</serviceProvider>
        <taskingAreaID>Planning area 1</taskingAreaID>
        <statusID>Planned</statusID>
        <acquisitionParameters>
          <platform>ENVISAT</platform>
          <sensorType>RADAR</sensorType>
          <operationalMode>ASA_WS_OP</operationalMode>
          <polarisationMode>S</polarisationMode>
          <polarisationChannels>HH</polarisationChannels>
          <antennaLookDirection>LEFT</antennaLookDirection>
          <processingLevel>1A</processingLevel>
          <acquisitionLength>400.05965042629</acquisitionLength>
          <acquisitionArea>20000</acquisitionArea>
          <notificationType>Short</notificationType>

          <!-- Area of the footprint, Km2 to be calculated by the POR -->
        </acquisitionParameters>
        <serviceType>
          <serviceTypeName></serviceTypeName>
          <serviceTypeDescription>Default CSN Service</serviceTypeDescription>
          <!-- String, Indicative text of the selected service type -->
          <serviceTypeSatelliteOperator>LICENSE</serviceTypeSatelliteOperator>
          <!--String, Comma separated values of Service Identifiers(with at least
one value), can be: LICENSE,DOWNLINK, IMAGE_PROCESSING,IMAGE_DELIVERY -->

          <additionalServiceElementsSO>EMERGENCY,ON_BOARD_RECORDER</additionalServiceElementsSO>
          <!--String, Comma separated values of extra Fee Identifiers, can be:
EMERGENCY,ON_BOARD_RECORDER,ARCHIVE -->

          <serviceTypeServiceProvider>DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION</servic
eTypeServiceProvider>
          <!--String, Comma separated values of Service Identifiers (with at least
one value), can be:
DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION,VESSEL_DETECTION,ACTIVITY_DETECTION -->
          <additionalServiceElementsSP></additionalServiceElementsSP>
          <!--String, Comma separated values, initially empty, could be expanded
in the future -->

          <maxDownlinkDelay>1.5</maxDownlinkDelay>
          <!-- float in minutes the delay allowed for downlink, cannot be null, if no
value set to 0.0 -->

          <deliveryDelaySatelliteOperator>2.5</deliveryDelaySatelliteOperator>

```

```

satellite operator (Provider 1)-->      <!-- A delay in decimal hours acceptable for the NRT processing for the
                                         <deliveryDelayServiceProvider>1.5</deliveryDelayServiceProvider>
service provider (Provider 2)-->      <!-- A delay in decimal hours acceptable for the
                                         <shortNotificationThreshold>72</shortNotificationThreshold>
                                         </serviceType>
                                         </Scene>
                                         </Scenes>
</TASK>

```

</TaskMessage>Figure 5-32 Example for the message type *TaskMessage*, sent when the PO starts a tasking

```

<?xml version="1.0" encoding="UTF-8"?>
<SceneMessage>
  <MessageID>2013-09-30T17:21:56+00:00</MessageID>
  <Scene>
    <project>CleanSeaNet</project>
    <ServiceType>Service Unit</ServiceType>
    <SensorMode>ASA_WS_OP</SensorMode>
    <Platform>ENVISAT</Platform>
    <Polarization>VV</Polarization>
    <usageofssr>>false</usageofssr>
    <FrameID>19644</FrameID>
    <TaskingAreaID>Planning area 5</TaskingAreaID>
    <TaskingType>0</TaskingType>
    <ServiceProviderID>EGEOS</ServiceProviderID>
    <LicenseProviderID>ESA</LicenseProviderID>
    <NotificationType>Short</NotificationType>
    <CustomerID>EMSA</CustomerID>
    <AcquisitionTime>2012-03-08T08:20:12.460Z</AcquisitionTime>
    <AcquisitionLength>562.13849803064</AcquisitionLength>
    <StatusID>Allocated</StatusID>
  </Scene>
</SceneMessage>

```

Figure 5-33 Example of message sent when a scene is allocated, *SceneMessage*

```

<?xml version="1.0" encoding="UTF-8"?>
<TaskMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="POR_JOU_FINSYS_v1.3.xsd">
  <MessageID>2011-03-24T18:21:21+01:00</MessageID>
  <TASK>
    <Identifier>9996</Identifier>
    <TaskFormNumber>1</TaskFormNumber>
    <TaskStatus>Planned</TaskStatus>
    <Description></Description>
    <Scenes>
      <Scene>
        <frameID>9911328</frameID>
        <project>PROJECT_1</project>
        <beginPosition>2012-07-20T20:03:28.380Z</beginPosition>
        <endPosition>2012-07-20T20:03:28.380Z</endPosition>

        <satelliteOperator>ESA</satelliteOperator>
      </Scene>
    </Scenes>
  </TASK>
</TaskMessage>

```



```

<serviceProvider>Ksat</serviceProvider>
<taskingAreaID>Planning area 1</taskingAreaID>
<statusID>Allocated</statusID>
<acquisitionParameters>
  <platform>ENVISAT</platform>
  <sensorType>RADAR</sensorType>
  <operationalMode>ASA_WS__OP</operationalMode>
  <polarisationMode>S</polarisationMode>
  <polarisationChannels>HH</polarisationChannels>
  <antennaLookDirection>LEFT</antennaLookDirection>
  <processingLevel>1A</processingLevel>
  <acquisitionLength>400.05965042629</acquisitionLength>
  <acquisitionArea>20000</acquisitionArea>
  <notificationType>Short</notificationType>

  <!-- Area of the footprint, Km2 to be calculated by the POR -->
</acquisitionParameters>
<serviceType>
  <serviceTypeName></serviceTypeName>
  <serviceTypeDescription>Default CSN Service</serviceTypeDescription>
  <!-- String, Indicative text of the selected service type -->
  <serviceTypeSatelliteOperator>LICENSE</serviceTypeSatelliteOperator>
  <!--String, Comma separated values of Service Identifiers(with at least
one value), can be: LICENSE,DOWNLINK, IMAGE_PROCESSING,IMAGE_DELIVERY -->

  <additionalServiceElementsSO>EMERGENCY,ON_BOARD_RECORDER</additionalServiceElementsSO>
  <!--String, Comma separated values of extra Fee Identifiers, can be:
EMERGENCY,ON_BOARD_RECORDER,ARCHIVE -->

  <serviceTypeServiceProvider>DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION</service
eTypeServiceProvider>
  <!--String, Comma separated values of Service Identifiers (with at least
one value), can be:
DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION,VESSEL_DETECTION,ACTIVITY_DETECTION -->
  <additionalServiceElementsSP></additionalServiceElementsSP>
  <!--String, Comma separated values, initially empty, could be expanded
in the future -->

  <maxDownlinkDelay>1.5</maxDownlinkDelay>
  <!-- float in minutes the delay allowed for downlink, cannot be null, if no
value set to 0.0 -->

  <deliveryDelaySatelliteOperator>2.5</deliveryDelaySatelliteOperator>
  <!-- A delay in decimal hours acceptable for the NRT processing for the
satellite operator (Provider 1)-->

  <deliveryDelayServiceProvider>1.5</deliveryDelayServiceProvider>
  <!-- A delay in decimal hours acceptable for the NRT processing for the
service provider (Provider 2)-->

  <shortNotificationThreshold>72</shortNotificationThreshold>
</serviceType>
</Scene>
</Scenes>
</TASK>

```

Figure 5-34 Example for the message type *TaskMessage*, sent when the PO starts approval

```

<?xml version="1.0" encoding="UTF-8"?>
<TaskInfoList>
  <Year>2012</Year>
  <Month>03</Month>

```

```

<TaskIdentifier>989</TaskIdentifier>
<TaskLabel>t</TaskLabel>
<POR_url>http://twls11/group/cleanseanet/planning</POR_url>
<ApprovedByFO>false</ApprovedByFO>
<NameAO></NameAO>
  <NameFO></NameFO>
  <NamePO>SD_1 -</NamePO>
<UserNameAO></UserNameAO>
<ApprovalDeadline>2013-10-03</ApprovalDeadline>
<ProviderList>
  <Provider>EGEOS</Provider>
</ProviderList>
<MailListTo>
  <Email>csndc-test@acsys.it</Email>
</MailListTo>
<MailListCC>
  <Email>csndc-test@acsys.it</Email>
</MailListCC>
</TaskInfoList>

```

Figure 5-35 Example of *TaskInfoList*, sent when the PO starts approval

```

<?xml version="1.0" encoding="UTF-8"?>
<TaskInfoList>
  <Year>2012</Year>
  <Month>03</Month>
  <TaskIdentifier>989</TaskIdentifier>
  <TaskLabel>t</TaskLabel>
  <POR_url>http://twls11/group/cleanseanet/planning</POR_url>
  <ApprovedByFO>true</ApprovedByFO>
  <NameAO>AO_UP05_01 AO_UP05_01</NameAO>
    <NameFO>CSN Financial Officer -</NameFO>
    <NamePO></NamePO>
  <UserNameAO>AO_UP05_01</UserNameAO>
  <ApprovalDeadline>2013-10-03</ApprovalDeadline>
  <ProviderList>
    <Provider>EGEOS</Provider>
  </ProviderList>
  <MailListTo>
    <Email>csndc-test@acsys.it</Email>
  </MailListTo>
  <MailListCC>
    <Email>csndc-test@acsys.it</Email>
  </MailListCC>
</TaskInfoList>

```

Figure 5-36 Message of type *TaskInfoList*, example sent when the FO approves an order

```

<?xml version="1.0" encoding="UTF-8"?>
<ReportInfo>
  <Year>2012</Year>
  <Month>03</Month>
  <TaskIdentifier>989</TaskIdentifier>
  <TaskLabel>t</TaskLabel>
  <Provider>ESA</Provider>
  <NameAO>CSN Authorising Officer -</NameAO>

```

```

<NameFO></NameFO>
<NamePO>SD_1 -</NamePO>
<UserNameAO>AO</UserNameAO>
  <MailListTo>
    <Email>csndc-test@acsys.it</Email>
  </MailListTo>
  <MailListCC> <!-- List of mails to put on the CC Field -->
    <Email>ao@emsa.europa.eu_test</Email>
<Email>csndc-test@acsys.it</Email>
  </MailListCC>
</ReportInfo>

```

Figure 5-37 Example of ReportInfo message generated when an order is approved

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2008 rel. 2 (http://www.altova.com) by mazuki (darksiderg) -->
<CSNMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="POR_JOU_FINSYS_v1.4.xsd">
  <timestamp>2010-02-26T12:54:37Z</timestamp>
  <sender>POR</sender>
  <newOrder>
    <frameID>9998</frameID>
    <project>FRONTEX</project>
    <eopIdentifier>rs2_16-99991-1</eopIdentifier>
    <beginPosition>2002-11-17T10:44:00Z</beginPosition>
    <endPosition>2002-11-17T10:45:38Z</endPosition>
    <satelliteOperator>ESA</satelliteOperator>
    <serviceProvider>Ksat</serviceProvider>
    <acquisitionParameters>
      <platform>satellite1</platform>
      <sensorType>RADAR</sensorType>
      <!-- This can be:RADAR,OPTICAL-->
      <operationalMode>ASA_WSM</operationalMode>
      <polarisationMode>S</polarisationMode>
      <!-- Can be: S,D,T, Q,UNDEFINED, meaning respectively Single, Dual, Twin, Quad -->
      <polarisationChannels>HH</polarisationChannels>
      <antennaLookDirection>LEFT</antennaLookDirection>
      <!-- This can be:LEFT,RIGHT -->
      <processingLevel>1A</processingLevel>
      <!-- This can be:1A,1B,2,3-->
      <acquisitionLength>234</acquisitionLength>
      <!-- Length of the footprint in Km to be calculated by the POR-->
      <acquisitionArea>20000</acquisitionArea>
      <!-- Area of the footprint, Km2 to be calculated by the POR-->
      <notificationType>Standard</notificationType>
      <!-- Can be Standard, Medium, Short Notification, automatically computed by the POR-->
    </acquisitionParameters>
    <serviceOrder>
      <serviceOrderID>9001</serviceOrderID>
      <correspondingLicenseID>9001</correspondingLicenseID>
    </serviceOrder>
  </newOrder>
</CSNMessage>

```

```

        <taskingAreaID>Portugal</taskingAreaID>
    </serviceOrder>
    <serviceType>
        <serviceTypeName/>
        <serviceTypeDescription>Default CSN Service</serviceTypeDescription>
        <!-- String, Indicative text of the selected service type -->
        <serviceTypeSatelliteOperator>LICENSE</serviceTypeSatelliteOperator>
        <!--String, Comma separated values of Service Identifiers(with at least one value), can be:
LICENSE,DOWNLINK, IMAGE_PROCESSING,IMAGE_DELIVERY -->

        <additionalServiceElementsSO>EMERGENCY,ON_BOARD_RECORDER</additionalServiceElementsSO>
        <!--String, Comma separated values of extra Fee Identifiers, can be:
EMERGENCY,ON_BOARD_RECORDER,ARCHIVE -->

        <serviceTypeServiceProvider>DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION</serviceTypeServiceProvider>
        <!--String, Comma separated values of Service Identifiers (with at least one value), can be:
DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION,VESSEL_DETECTION,ACTIVITY_DETECTION -->
        <additionalServiceElementsSP/>
        <!--String, Comma separated values, initially empty, could be expanded in the future -->
        <maxDownlinkDelay>1.5</maxDownlinkDelay>
        <!-- float in minutes the delay allowed for downlink, cannot be null, if no value set to 0.0 -->
        <deliveryDelaySatelliteOperator>2.5</deliveryDelaySatelliteOperator>
        <!-- A delay in decimal hours acceptable for the NRT processing for the satellite operator
(Provider 1)-->

        <deliveryDelayServiceProvider>1.5</deliveryDelayServiceProvider>
        <!-- A delay in decimal hours acceptable for the NRT processing for the service provider
(Provider 2)-->

        <shortNotificationThreshold>72</shortNotificationThreshold>
    </serviceType>
</newOrder>
<newOrder>
    <frameID>9999</frameID>
    <project>FRONTEX</project>
    <eopIdentifier>rs2_16-99991-2</eopIdentifier>
    <beginPosition>2002-11-17T10:44:00Z</beginPosition>
    <endPosition>2002-11-17T10:45:38Z</endPosition>
    <satelliteOperator>ESA</satelliteOperator>
    <serviceProvider>CLS</serviceProvider>
    <acquisitionParameters>
        <platform>satellite2</platform>
        <sensorType>RADAR</sensorType>
        <!-- This can be:RADAR,OPTICAL-->
        <operationalMode>ASA_WSM</operationalMode>
        <polarisationMode>S</polarisationMode>
        <!-- Can be: S,D,T, Q,UNDEFINED, meaning respectively Single, Dual, Twin, Quad -->
        <polarisationChannels>HV</polarisationChannels>
        <antennaLookDirection>RIGHT</antennaLookDirection>
        <!-- This can be:LEFT,RIGHT -->

```

```

        <processingLevel>1B</processingLevel>
        <!-- This can be:1A,1B,2,3-->
        <acquisitionLength>234</acquisitionLength>
        <!-- Length of the footprint in Km to be calculated by the POR-->
        <acquisitionArea>30000</acquisitionArea>
        <!-- Area of the footprint, Km2 to be calculated by the POR-->
        <notificationType>Standard</notificationType>
        <!-- Can be Standard, Medium, Short Notification, automatically computed by the POR-->
    </acquisitionParameters>
    <serviceOrder>
        <serviceOrderID>9001</serviceOrderID>
        <correspondingLicenseID>9001</correspondingLicenseID>
        <taskingAreaID>Portugal</taskingAreaID>
    </serviceOrder>
    <serviceType>
        <serviceTypeName/>
        <serviceTypeDescription>Default CSN Service</serviceTypeDescription>
        <!-- String, Indicative text of the selected service type -->
        <serviceTypeSatelliteOperator>LICENSE</serviceTypeSatelliteOperator>
        <!--String, Comma separated values of Service Identifiers(with at least one value), can be:
LICENSE,DOWNLINK, IMAGE_PROCESSING,IMAGE_DELIVERY -->
        <additionalServiceElementsSO/>
        <!--String, Comma separated values of extra Fee Identifiers, can be:
EMERGENCY,ON_BOARD_RECORDER,ARCHIVE -->

        <serviceTypeServiceProvider>IMAGE_PROCESSING,OIL_SPILL_DETECTION</serviceTypeServiceProvider>
        <!--String, Comma separated values of Service Identifiers (with at least one value), can be:
DOWNLINK,IMAGE_DELIVERY,IMAGE_PROCESSING,OIL_SPILL_DETECTION,VESSEL_DETECTION,ACTIVITY_DETECTION -->
        <additionalServiceElementsSP/>
        <!--String, Comma separated values, initially empty, could be expanded in the future -->
        <maxDownlinkDelay>1.5</maxDownlinkDelay>
        <!-- float in minutes the delay allowed for downlink, cannot be null, if no value set to 0.0 -->
        <deliveryDelaySatelliteOperator>2.5</deliveryDelaySatelliteOperator>
        <!-- A delay in decimal hours acceptable for the NRT processing for the satellite operator
(Provider 1)-->

        <deliveryDelayServiceProvider>1.5</deliveryDelayServiceProvider>
        <!-- A delay in decimal hours acceptable for the NRT processing for the service provider
(Provider 2)-->

        <shortNotificationThreshold>72</shortNotificationThreshold>
    </serviceType>
</newOrder>
</CSNMessage>

```

Figure 5-38 Example of NewOrder message generated when a tasking is completed (approved by AO)

5.8 JOURNALING

The sequence of actions underlying the process of journaling is exemplified in the following sequence diagrams.

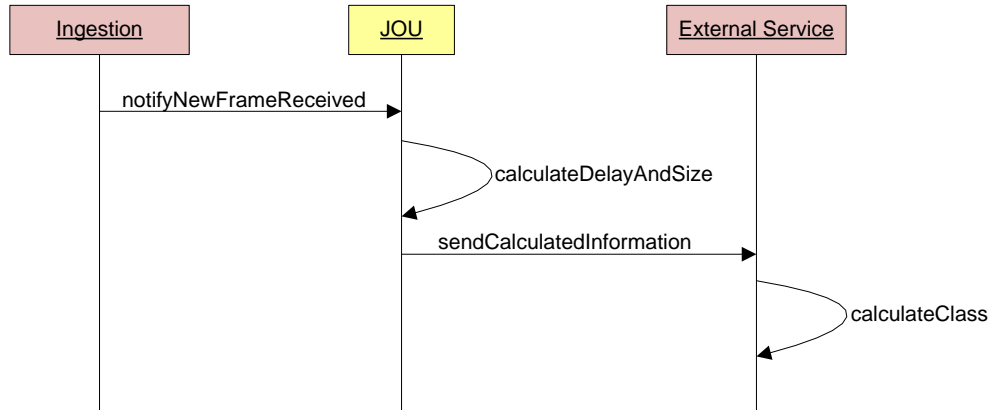


Figure 5-39 - Dynamic Flow of Journaling information process

Whenever a new frame is received by Ingestion, the later notifies the JOURNAL about the reception. On its side, JOURNAL calculates the delay and image size for the frame and sends the calculated information to an external service capable of calculating the class.

This process runs for every new frame, and the data related to the frame is stored for future reporting.

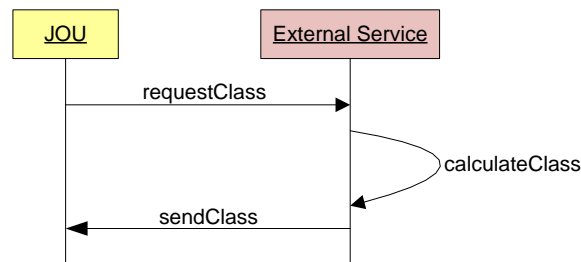


Figure 5-40 - Dynamic Flow of class request

Whenever a user requests a report, the JOURNAL must ask the external service for the classes of all frames displayed on the report. This request is done in batch, thus calling the external service only once to get all the classes to display.

The external service sends the requested classes back to the JOURNAL.

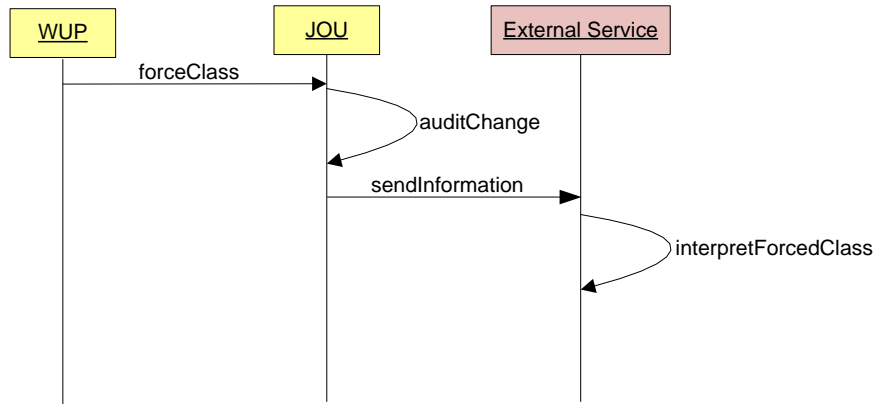


Figure 5-41 - Dynamic Flow of force class request

It is possible for a user to force a certain class on a frame by adding a set of parameters and passing them to the JOU. The JOU then store audit information about the requested change and send the new information to the external service.

The external service then interprets the new information and the forced class to provide the invoicing.

5.9 COM

All processes relevant to COM are controlled by Liferay, thus, only example diagrams will be presented in this chapter.

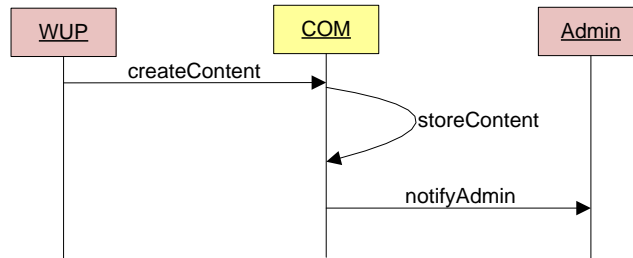


Figure 5-42 - Dynamic Flow of create Forum content

Whenever a forum user creates new content, i.e. posts a new thread/topic, replies to a thread/topic, the data created will be passed to the COM for storage. After the data is stored, the COM informs the forum administrator about the new content.

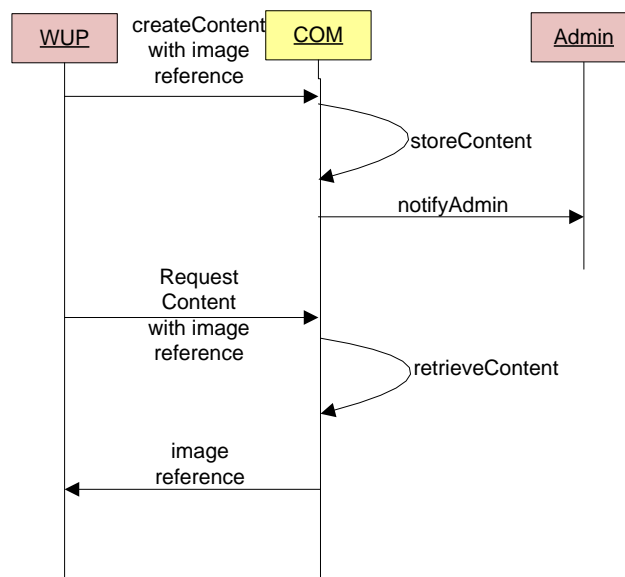


Figure 5-43 - Dynamic Flow of create Forum content from an EO product

When a user is viewing an EO product (image) he can create a new thread on the forum from that image. A reference to the image is maintained in order for the user to be able to return to the image from the forum.

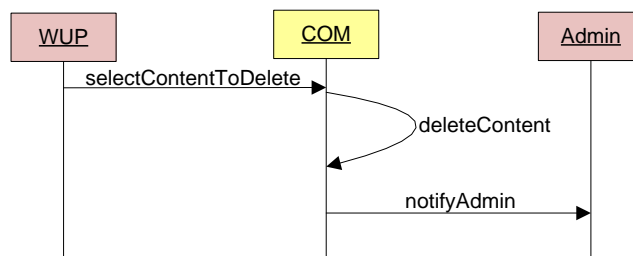


Figure 5-44 - Dynamic Flow of delete WIKI content

Users can delete WIKI content they created, or in case the user is an administrator, he can delete other users content. This is done by selecting the content to delete. The selected content is then passed to the COM to be deleted from storage and the administrator is informed about the event.

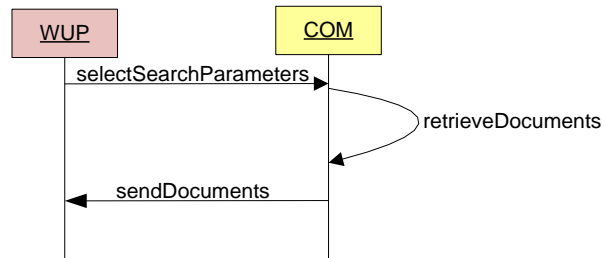


Figure 5-45 - Dynamic Flow of search documents

When searching for documents, a user selects a set of parameters and then passes them to the COM. The COM then processes the parameters and retrieves all documents corresponding to the selected parameters and sends them back to the user.

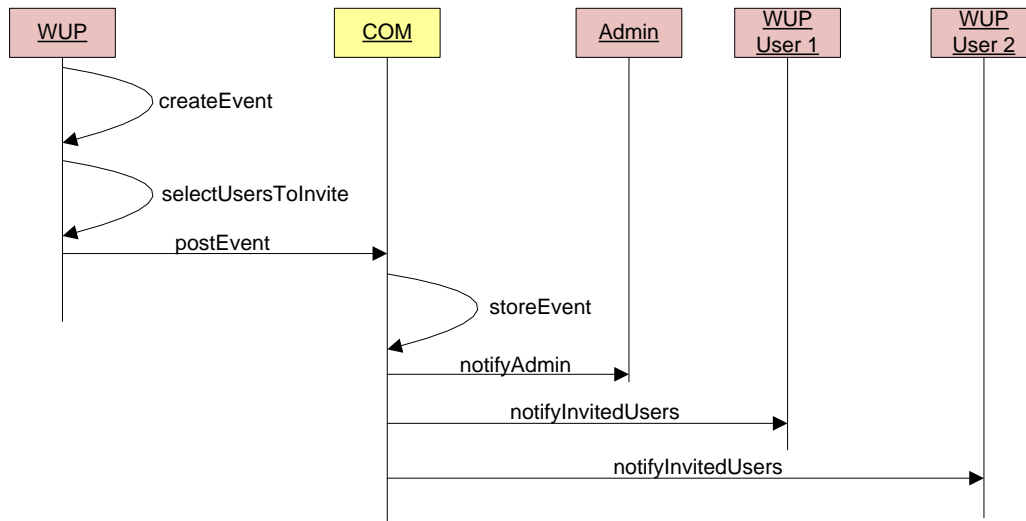


Figure 5-46 - Dynamic Flow of post calendar event

To create a calendar event, a user starts by filling the event details. He then chooses other users to invite to the event and finally posts the new event to the COM. The COM stores the event and notifies the administrator about the new event. Finally, the COM notifies all invited users about the new event.

In all possible scenarios, community data needs to be protected in a way that the system cannot allow the input of invalid data, thus preventing denial of service attacks. It must also avoid cross site scripting or other types of attacks. To achieve this goal, all user input will be validated.

5.10 Oil Spill Models integration and workflow

Oils spill models integration deserves a dedicated section, because it is a specific type of integration of an external components which spans across various sub-components of the CSNDC.

There are 2 types of workflows:

- Manual workflow
- Automatic workflow

For the release 1.7 the manual workflow was implemented. The automatic workflow is mostly identical to the manual workflow, with the only difference that the triggering is automatic.

The architecture of the oil spill models manual integration is composed of the following elements:

- A GUI from which to trigger the oil spill model with the appropriate parameters and to display the processing results
- The core oil spill model processor which follows the WPS (*Web Processing Service*) specification
- The Oil Spill WFS service, used by the Oil Spill WPS to retrieve the oil spill feature involved
- The External FTP server used by the WPS model to store the processing results and by the ingestion to import them into CSNDC
- An ingestion component which follows the same logic as the ingestion of all other data types (see § 5.2)

The component diagram is illustrated in the next figure (a different colour has been used to depict the Oil Spill WPS which is an external component).

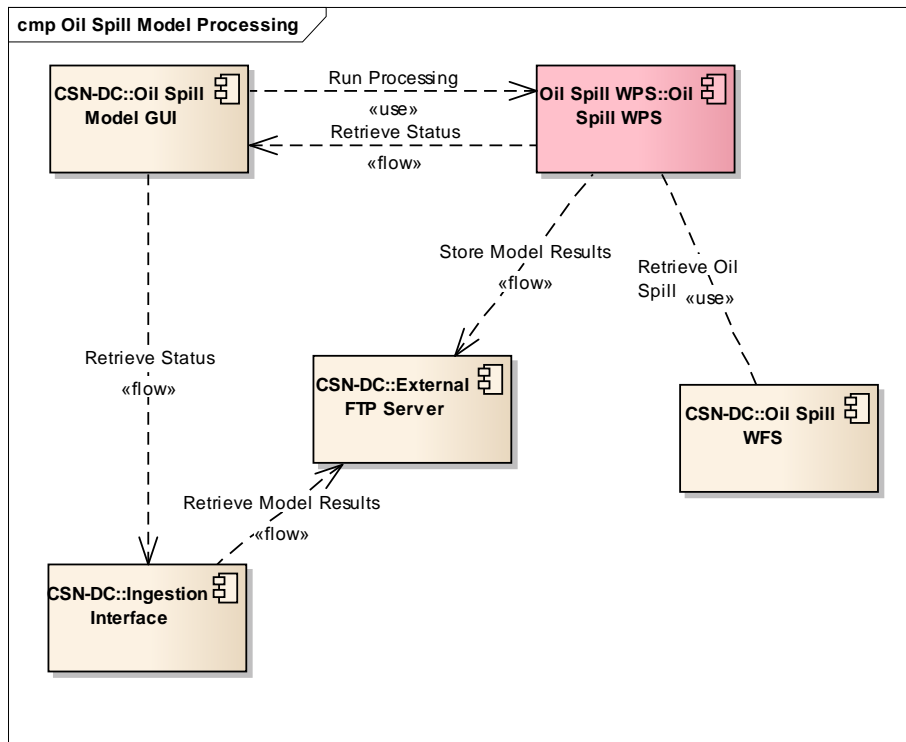


Figure 5-47 – Component diagram illustrating the elements involved in the Oil Spill model processing

The workflow for the processing is represented in the following sequence diagram and summarised as follows (the Oil Spill WPS, being external to the CSNDC system is reported with a different colour):

- The user accesses the oil spill GUI through the WUP
- The user defines the oil spill execution parameters and triggers the oil spill model
- The oil spill model GUI prepares a call to the Oil Spill Model WPS, providing the parameters as an XML (see example in § 14) and implements the actual WPS request to the model provider
- the Oil Spill WPS following the WPS protocol returns a monitoring URL, that can be polled regularly to get the status of the oil spill model processing
- The Oil Spill WPS retrieves the oil spill reported in the request via WFS using the Oil Spill WFS service
- the oil spill model GUI routinely gets the status of the model and displays it on the WUP GUI
- independently on the polling process described above, the Oil Spill WPS, upon successful completion of the model run stores the results on the External FTP Server (using the credentials dedicated to the oil spill model)
- in parallel the Ingestion Interface, routinely polls the External FTP Server (including the account corresponding to the oil spill model subscribed) and if finds the result of an oil spill models stores them internally to the CSNDC
- in parallel the GUI of the WUP polls the ingestion interface, to get the status of the storage and upon successful storage reports the successful completion of the model end to end process, meaning that the model data are available to be displayed

From the point of view of the data model for the results of the oil spill prediction, the data model is exactly equal to the data model of the detected oil spills. The only difference will be in the values of the field **OilSpillType/origin** (§ 8.1.2) which will have the value **PREDICTED**.

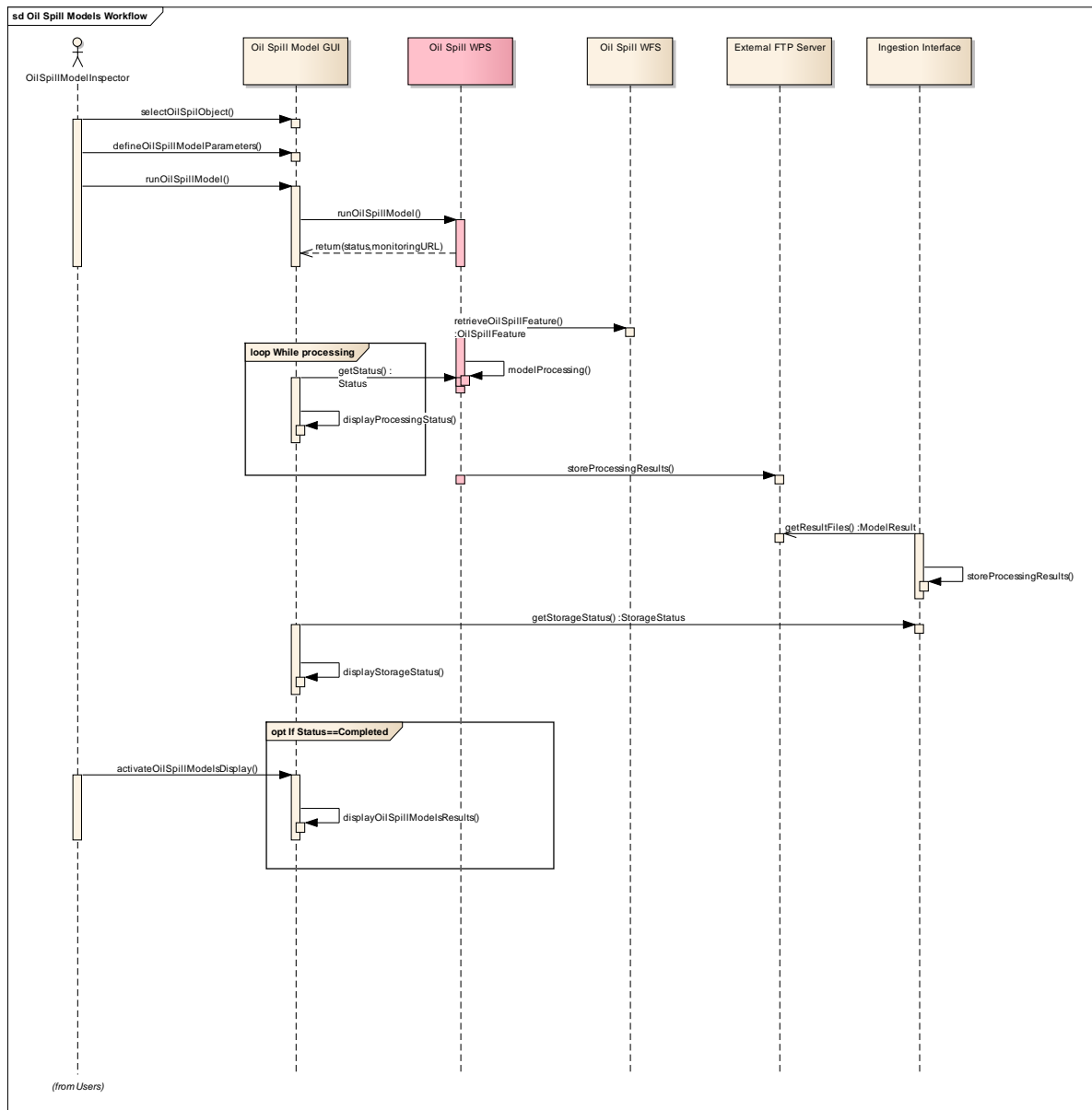


Figure 5-48 – Oil spill modelling execution workflow

5.11 Archive and Restore Tools

5.11.1 Overview of the functionality

A number of tools were implemented to allow manual and automatic archive and restore of CSNDC data. This is necessary to have the CSNDC aligned with the general EMSA Data Policy, which dictates that as a default data older than 18 months shall be archived and can be restored on demand.

The high level requirements are here summarised:

- It shall be possible to archive and restore the following data types:
 - Data packages received from the Service Provider (e.g. EOP, OSN, etc.)
 - AIS data
 - Geoserver layers
- The archive and restore functions shall be provided via Web Services
- It shall be possible to run the Web Services using a script
- It shall be possible to encapsulate this script into a cron job that is executed routinely on the system (this is a basic operating system function implemented independently from the CSNDC system)

This set of tools implements a functionality that is referred to as *CSN-DataPolicy*.

The CSN-DataPolicy shall implement its functionalities as a web service (CSN-DataPolicyService), and a script based client (CSNDC-DataPolicyClient). Daily based the CSNDC-DataPolicyClient shall invoke the CSN-DataPolicyService in order to provide the following functionalities:

1. **Automatic Archiving.** "CSN-DataPolicyService.Archive (numberofdays)":
 - a. move all the packages linked to an acquisition older than "numberofdays" (EOP, OSW, QNO, DER, QUA, etc...) FROM the ISM light server TO the EMSA's off-line archive service, and export from the CSNDC-WFS the AIS data and save them also on the EMSA's off-line archive service. By default "numberofdays" is the number of day equivalent to 18 months.
 - b. Remove the layers from the CSNDC-WMS linked to the EOP products older than the "numberofdays".
 - c. Remove the AIS data from the CSNDC-WFS older than the "numberofdays".
 - d. Set the status as "archived" of the EOP metadata products catalogued in the CSNDC-CSW older than the "numberofdays". The CSNDC-CSW metadata need to be updated with the information to retrieve the CSN's datasets (packages and AIS) from the EMSA off-line archive service.
 - e. When a CSNDC user selects an "archived" product through a CSNDC client (i.e. CSNDC-GISViewer), the CSNDC client shall visualize only the bounding box of the image and be informed that the selected acquisition is archived, therefore no longer available on-line and that, it is possible to restore the data by issuing a request to the EMSA MSS.

2. **On demand Archiving.** “CSN-DataPolicyService.Archive (ProductsIDs)”:
 - a. The CSNDC-DataPolicyClient can invoke the CSN-DataPolicyService specifying the EOP product identification that needs to be archived. Then steps 1.a, 1.b, 1.c, and 1.d have to be applied.
3. **Restoring from archive.** “CSN-DataPolicyService.Restore (ProductsIDs, numberofdays)”.
Through the CSNDC-DataPolicyClient the CSN-DataPolicyService can be invoked specifying the products identification (one or more than one) to restore:
 - a. move all the packages linked the products identification (EOP, OSW, QNO, DER, QUA, etc...) FROM EMSA’s off-line archive service TO the ISM light server.
 - b. change the status in “restored” in the CSNDC-CSW of all EOP product restored.
 - c. re-create the layers in the CSNDC-WMS associated to the EOP product to restore.
 - d. re-feed the CSNDC-WFS with the AIS data archived in the EMSA’s off-line archive service and associated to the EOP product to restore.
 - e. The restored EOP packages, layers and AIS data will be available on-line for a number of days specified by call to the CSN-DataPolicyService and then automatically archived again.
4. **Turning on/off.** It shall be possible to turn ON or OFF the automatic archiving functionality of the CSNDC-DataPolicy.

5.11.2 Implementation logic

In practical terms of the following functions shall be implemented during archiving:

- Determine the data to be archived
- Remove physically data from the file system and copy them into the archive location (typically this is a shared file system, but it could be an FTP site, etc.).
- Move AIS data from the online database table into the offline database table. Online is the table which is queried by the GIS Viewer and which exposes data through WFS, therefore when data are moved from the online table into the offline table, they are no longer accessible using these interfaces
- Removing from Geoserver the relevant layers, so that they are not part of the layers made available by WMS (e.g. returned by the GetCapabilities request)
- Set the status of the relevant scenes to ARCHIVED, so that this is visible both in the GIS Viewer (when the user is searching for data) and when submitting a CSW request

During the restore operation, basically the reverse operations shall be carried out, i.e.:

- Determine the data to be restored
- Copying physical files back from the archive location into the online location (ISM): NOTE when restoring archived data, the archived copy is not removed, but it is kept. This means that if data are archived and restored again, a double copy of the data files is present (online and offline). A subsequent archive request will simply remove the physical data from

the online location, relying on the fact that the data are already present in the backup location.

- Moving AIS data from the offline table into the online table (making them again accessible through GIS viewer and WFWS interfaces)
- Set the status of the relevant scenes to DELIVERED, so that this is visible both in the GIS Viewer (when the user is searching for data) and when submitting a CSW request

The granularity of the archive / restore process is the individual service ID. So, once the list of service IDs to be archived / restored have been determined, all relevant steps are executed for all data associated to this service ID.

Some more details are provided here regarding the individual macro steps.

Here follows a component diagram of the archive/restore functionality. the specific components developed for this functionality are:

- ArchiveRestoreClient: a shell script
- ArchiveRestoreWS: a web service implemented in PHP

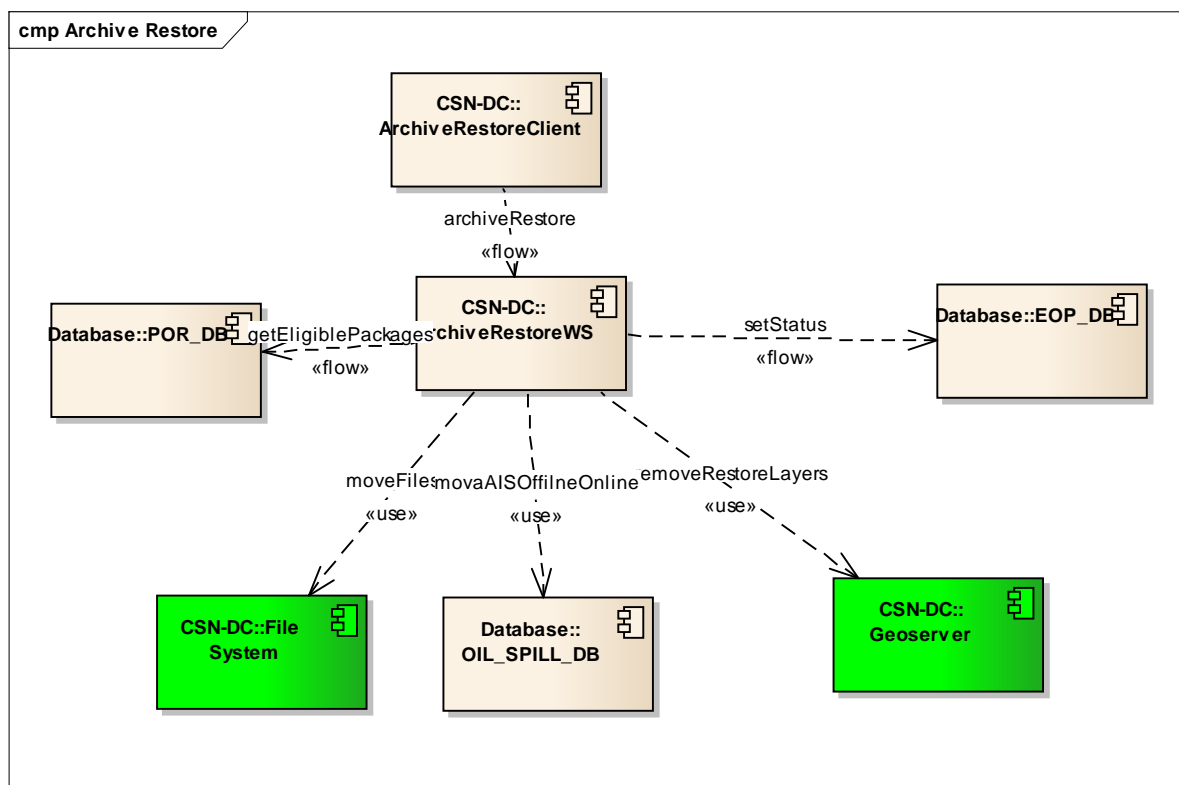


Figure 5-49 – Archive / restore functionality component diagram

The workflow is implemented by the archiveRestoreWS, a web service which may be called directly as such, or via a specific client (ArchiveRestoreClient), which is callable via command line (and it is basically a wrapper around the WS). The client for example can be invoked by a cron job and performed on a daily basis at predefined times.

The dynamic behaviour of the archive process is depicted in the following figure, which also explains the dependencies between packages illustrated above.

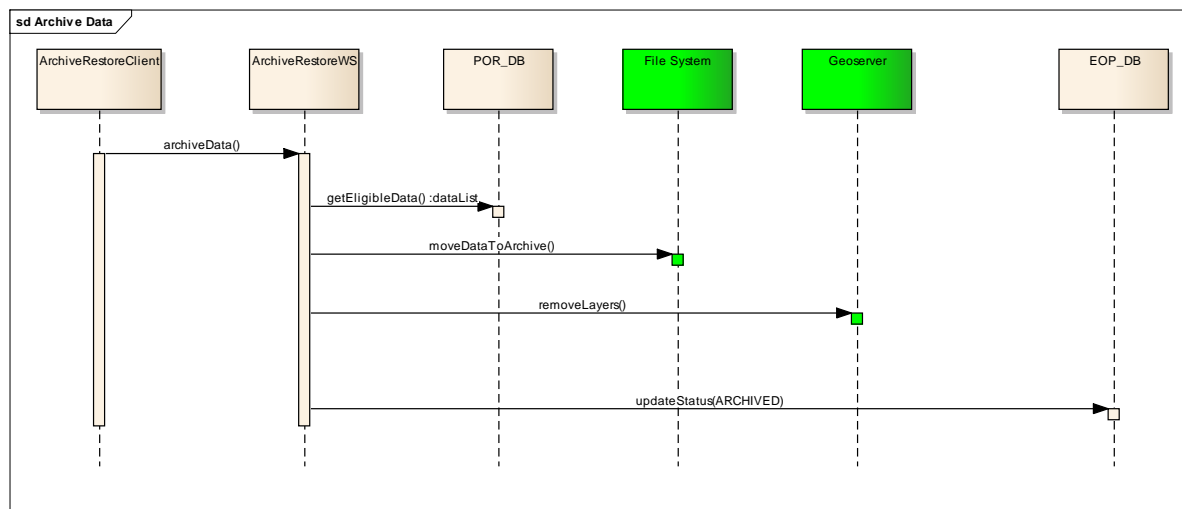


Figure 5-50 – Archiving process

The main steps are illustrated briefly here (while the details are reported in the following sections):

- ArchiveRestoreClient triggers the ArchiveRestoreWS, passing the necessary parameters (see below)
- the ArchiveRestoreWS retrieves the list of eligible files (i.e. files to be archived) from the POR_DB
- the ArchiveRestoreWS moves physically files from the file system into the archiving storage
- the ArchiveRestoreWS removes the corresponding layers (e.g. TIFF images and SAR derived data) from geoserver
- the ArchiveRestoreWS updates the status of the corresponding data into the EOP_DB (e.g. setting to ARCHIVED)

A similar sequence is performed for restoring the data, as reported below.

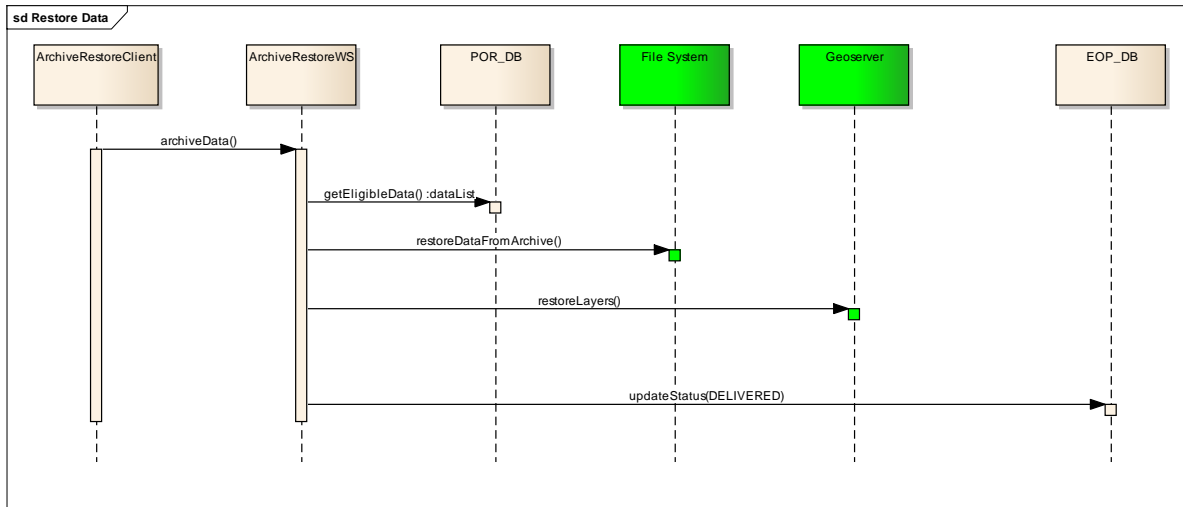


Figure 5-51 – Restoring process

The main steps are illustrated briefly here (while the details are reported in the following sections):

- ArchiveRestoreClient triggers the ArchiveRestoreWS, passing the necessary parameters (see below)
- the ArchiveRestoreWS retrieves the list of eligible files (i.e. files to be restored) from the POR_DB
- the ArchiveRestoreWS moves physically files from the archiving storage to the online location
- the ArchiveRestoreWS restores the corresponding layers (e.g. TIFF images and SAR derived data) from geoserver
- the ArchiveRestoreWS updates the status of the corresponding data into the EOP_DB (e.g. setting to DELIVERED)

5.11.2.1 Determine data to be archived

As input to the archive function it is possible to provide either of the following:

- A number of days
- A list of service IDs

The data to be archived are determined either by using the individual service IDs or by checking the date when products were originally catalogue. In particular, when the number of days are provided in the function call, the following query is executed (e.g. setting 30 as number of days)

```
SELECT order_detail_id, filename FROM validity_checks vc WHERE (vc.published_datetime <= SYSDATE - 30) AND vc.order_detail_id NOT IN (SELECT order_detail_id FROM x_order_details_archive xoda WHERE
```

(xoda.archive_status_id <> 3) OR SYSDATE < (restore_datetime + number_of_days)) order by
vc.published_datetime

- In practice the following is checked: Data must have been published more than *number of days* ago
- Data must not be already in status archived
- In case data have been restored, the number of days defined when restoring, has already passed

5.11.2.2 Moving physical files from online to archive location

The physical file include the following files (.zip or .tgz):

- Packages, i.e.:
 - EOP
 - OSN
 - OSW
 - QNO
 - QUA
 - DER
 - PV
- AIS files: a set of GML files that include AIS data

In the on-line storage that packages are stored under the following path:

<ISM_BASE_DIR>/<SERVICE_GROUP>

Where:

- <ISM_BASE_DIR> is the base dir where the ISM store is defined, e.g. /shared_nfs/ism_store
- <SERVICE_GROUP> is a folder which is determined as CEIL(<SERVICE_ID>/100) and is 0-padded to the left to have a fixed size of 6 digits. Eg. Given service ID 5667, the folder will be 000057

The AIS data are placed into a folder named <SERVICE_ID>_AIS which will contain all GML data for that particular service. So it will be:

<ISM_BASE_DIR>/<SERVICE_GROUP>/<SERVICE_ID>_AIS

In the archive storage the path is slightly different, because it includes a subfolder for each service ID, like the following

<ARCHIVE_BASE_DIR>/<SERVICE_GROUP>/<SERVICE_ID>

And likewise for the AIS files they will be placed in a folder named as follows:

< ARCHIVE _BASE_DIR>/<SERVICE_GROUP>/<SERVICE_ID>/<SERVICE_ID>_AIS

The part highlighted in yellow indicates the difference in path between the online and archive storage. When the service ID to be archived are determined, the system also determines which are the exact list of files that shall be archived. These are all files listed in the PORUSR database in the VALIDITY_CHECKS table, where the status is Ok.

5.11.2.3 Moving AIS data from the online table into the offline table

The bulky AIS data are stored into a table named AISTRACKS for the DGRFTRUSR. During archive the relevant data, identified by the key ORDER_DATIL_ID matching the service ID to be archived, are removed from this table and copied into the AISTRACK_BKP table. The AISTRACK table is searched by the GIS Viewer and is configured to expose the data via WFS. Therefore when data are removed from the table they are automatically not exposed via neither the GIS Viewer, nor the WFS interfaces.

5.11.2.4 Removing Geoserver layers

The archive functions tries to remove the corresponding Geoserver layer, by using the Geoserver REST interfaces. In particular the following layers are tentatively removed (if existing):

- Raster layers corresponding to the PV images (EO image data)
- Raster layers corresponding to SAR Wind and Wave derived products

5.11.2.5 Setting the status to archived

The archive functions at the end of the process for a given service ID, sets the following in the CSW metadata:

- Sets the value of the status field to ARCHIVED
- Sets the archive location to the path where the data are physically archived (see § 5.11.2.2)
- Sets the archive date to the date time when this was executed

5.11.2.6 Determining data to be restored

The restore function allows to define a list of product IDs to be restored. Thus, the determination of eligible data is implicit. It also checks that the data have not been already restored, in which case, the system will not attempt to restore them again.

5.11.2.7 Copying physical files back from the archive location into the online location

This operations consists in copying the physical files which were archived as described in § 5.11.2.2 back into the original ISM storage. Data are not deleted from the archive location. A subsequent request to re-archive the data will simply consist in removing them from the online storage again.

5.11.2.8 Moving AIS data from the offline table into the online table

Data are copied back from the AISTRACK_BKP table into the AISTRACK table, simply using the ORDER_DETAIL_ID of the current service ID to be restored.

5.11.2.9 Setting the status to delivered

After successful restore of a given product ID, the status on the GIS Viewer is set to Delivered.

5.11.3 Implementation details and handling of non-nominal cases

The whole process of archive and restore is registered through a dedicated table, which is X_ORDER_DETIALS_ARCHIVE of the PORUSR, which includes the following columns:

- ORDER_DETAIL_ID: service ID
- ARCHIVE_STATUS_ID: status of the archiving, see below for more info
- ARCHIVE_DATETIME: time when archive was last executed
- RESTORE_DATETIME: time when restore was last executed
- ARCHIVE_LOCATION: location where the physical data are (it can be the archive or the restored path depending on the status of the data)
- ARCHIVE_NOTE: a summary of the archive/restore results

Here is an example of possible data written in this table:

OEDER_DETAIL_ID	ARCHIVE_STATUS_ID	ARCHIVE_DATETIME	RESTORE_DATE TIME	NUMBER_OF_DAYS	ARCHIVE_LOCATION	ARCHIVE_NOTE
134476	3	10-JUL-14 10:09:02.5349 52000	10-JUL-14 11:28:41.37580 9000	100	/shared_nfs/ism_store/001345	Restoring for 100 days Restored AIS files Restored files Created layer Warnings on restoring AIS data (see logs for details) Updated CSW metadata
134495	13	10-JUL-14 10:18:20.1766 39000	10-JUL-14 11:18:38.96912 9000	100	/shared_nfs/ism_store/001345	Restoring for 100 days Restored AIS files Warnings on restoring files (see logs for details) Created layer Warnings on restoring AIS data (see logs for details) Updated CSW

						metadata
134493	3	10-JUL-14 10:09:21.1844 63000	10-JUL-14 11:17:30.87392 0000	100	/shared_nfs/is m_store/00134 5	Restoring for 100 days Restored AIS files Restored files Created layer Warnings on restoring AIS data (see logs for details) Updated CSW metadata
134044	12	10-JUL-14 11:46:57.8436 04000	10-JUL-14 10:50:30.02379 3000	0	/shared_nfs/ar chive/001341/ 134044	Archiving data older then 30 days Warnings on removing layer (see logs for details) Archived files Archived AIS files Warnings on archiving AIS data (see logs for details) Updated CSW metadata
132292	1	10-JUL-14 11:46:53.3226 63000	10-JUL-14 10:22:25.39850 6000	0	/shared_nfs/ar chive/001323/ 132292	Archiving data older then 30 days Removed layer Archived files Archived AIS files Archived AIS data Updated CSW metadata

The status of archiving shall be carefully analysis considering that:

- Possible values of the ARCHIVE_STATUS_ID are:
 - 0 archiving
 - 1 archived (completed without errors and warnings)
 - 2 restoring
 - 3 restored (completed without errors and warnings)
 - 4 removed_layer
 - 5 archived_files
 - 6 archived_ais_files
 - 7 archived_ais_data
 - 8 restored_ais_files
 - 9 restored_files
 - 10 created_layer
 - 11 restored_ais_data
 - 12 archived_warnings (completed with warnings)
 - 13 restored_warnings (completed with warnings)
- Statuses 1 and 3 mean everything was ok. Status 12 and 13 means the process was completed with warnings (see below). All other statuses are intermediate and only occur when there is a critical error causing the operation to be aborted for the current service ID (the process continues with all the other service ID, in case there are any to be processed). The values of the intermediate statues (from 4 to 10) are in sequence, meaning that the previous were carried out.
- Any status between 4 and 10 means some unexpected error occurred making impossible to complete the operation for the current Service ID. This should never occur .
- The column ARCHIVE_NOTE indicates a synthetic summary of the problems/warnings occurred during the operation on the current service ID. For more details, it is necessary to analyse the log files.

The above table lists all service ID for which archive and/or restore was done (even if it was unsuccessful). It is therefore a sort of history of the archive / restore processes and is used by the system but can also be used for handling non-nominal situations.

Non-nominal situation may occur for the following possible causes (most likely):

- Trying to copy files:
 - The source file does not exist:
 - this may happen if there is a misalignment between the file system and the database
 - it can also happen that, if no AIS data exist for that service ID, the corresponding AIS data do not exist either
 - There is no sufficient space on the target destination (e.g. on the archive file system)
 - The system (typically it is the *apache* user, because the core service is a Web Service) does not have sufficient privileges to copy the file into the destination
- Trying to remove a file:
 - The source file does not exist: this may happen if there is a misalignment between the file system and the database
 - There is no sufficient space on the target destination (e.g. on the archive file system)
 - The system (typically it is the *apache* user, because the core service is a Web Service) does not have sufficient privileges to copy the file into the destination
- Trying to remove AIS data from the source database:
 - There were no AIS data for the current service ID
- Trying to remove geoserver layers:
 - The geoserver layer did not exist (this could be the result of an inconsistency between file system and geoserver configuration or because some additional layer, e.g. SAR Wind and Wave were never received in the CSNDC)

Some of these error causes may be related to non-problematic situation (e.g. if AIS data were not pre-existing, this is not necessarily an anomaly, but it could be that data for some reasons were not loaded, etc.). Conversely, some other error cases must be handled with more care, like for example monitoring if there is no disk space in the archive location, etc.

For this reason, the following approach was used for handling non-nominal cases:

- If there is any error in performing any of the sub-steps of the archive or restore, it is simply considered as a warning
- The process continues, trying to execute all other sub-steps
- All warnings are logged as described above, using:
 - A warning code in the database
 - A short warning description in the database
 - Detailed warning description in the log files

- If there are warnings processing a given service ID, the processing of other service IDs will continue independently (typically the causes of warning are specific of a given service ID, although there are global problems, e.g. disk space that may affect all service IDs)

The outcome of this policy is that even if a given sub-steps fails, the global process is considered finished. For example, even if the copy of some specific files, or the removal of a given geoserver layer fails, the process tries to continue to do the archiving and then leaves the option to the operator to monitor all non-nominal cases and decide on a case by case basis whether it deserves major attention, is some manual recovery action shall be taken or not.

5.12 Implementation of the Business Continuity Facility (BCF) support function

The CSN DC shall be capable to be operated on full redundancy equipment including local back-up servers working in switch-over or load-balancing between servers locally, as well as remote backup servers which should be ready to overtake crucial activities in case of disaster or serious malfunction in Primary Location. The EMSA Business Continuity Facility is located in Porto, Portugal. The system shall be updated such that the downtime should be less than one hour when switching from one location to the other. However even during downtime no data shall be lost.

In order to simplify the management of the various environments (test, pre-production and production), the synchronisation of data bases between environments shall be included.

The general environment and constraints are depicted in [E-V], namely section 2.5 of that document. In particular the ON/OFF model was chosen, where the Virtual Machines and the database are kept synchronised via infrastructure means (VM tools and Dataguard for the database). It is therefore assumed that the 2 environment will be continuously synchronised.

If this is true, the switch over should be easy to implement, as the 2 sites are always aligned.

The specific CSN-DC implementation therefore consisted in:

- analysis and fine tuning with EMSA to clarify the open points (see below)
- development (if needed) of scripts to re-configure some parameters (e.g. new URLs, end points, etc.)
- development of the scripts mentioned above to stop/start/check the system instance(s)

Basically the only reconfiguration to perform is referred to:

- database connection string
- CMAP end point

All the rest, assuming no physical IPs are used, but only the server names, will be automatically migrated.

The switch between BCF/EMSA and viceversa should be made with the following steps:

1. Stop the system instances
2. Configure different parameters

3. Start the system instances

Based on the EMSA constraints, the opemsa user needs to have permission to execute service start/stop command.

It's up to EMSA to configure /etc/sudoers and share the "opemsa" Public RSA authentication key between hosts. The configuration step needs to be executed by the root user on each host.

5.12.1 Start/stop scripts

This script will start/stop the whole CSN-DC system.

Usage

csndc-system.sh

./csndc-system.sh -h

Usage: ./csndc-system.sh [Options] {PROD|PREPROD|TEST} {start|stop|status}

This script check and start/stop the CSN-DC services.

Options:

-h : list available command line options (this page)

-v : verbose mode; shows additional information

Examples:

./csndc-system.sh PROD status

5.12.2 Configuration scripts

csndc-config.sh

./csndc-config.sh -h

./csndc-config.sh: option requires an argument - h

Usage: ./csndc-config.sh [Options] {prod|preprod|test} {emsa|bcf|acs}

It modifies parameter configurations on CLASS host, such as:

- WLS:
 - change Database server configuration on acs_global_config.ini
 - set JDBCConnection host on deegree configuration
 - set JDBCConnection host on vcat configuration
 - set Geoserver datastore host configuration and restart geoserver app on Weblogic
 - update MAP server url on Sinbad configuration
 - update MAP server url on csndc-report configuration
 - redeploy and restart csndc-report app on Weblogic
- PMAS:
 - change Database server configuration on acs_global_config.ini
 - update Database configuration on PDS

- PMAW:
 - change Database server configuration on `acs_global_config.ini`
 - update Database configuration on PDS

It is supposed that all the services have been stopped except WebLogic services and Database services. The script has to be run locally as root user on each host.

Options:

-h : list available command line options (this page)

-v : verbose mode; shows additional information

Examples:

`./csndc-config.sh prod bcf`

`./csndc-config.sh prod emsa`

5.12.3 Example of switch over between EMSA PROD and BCF

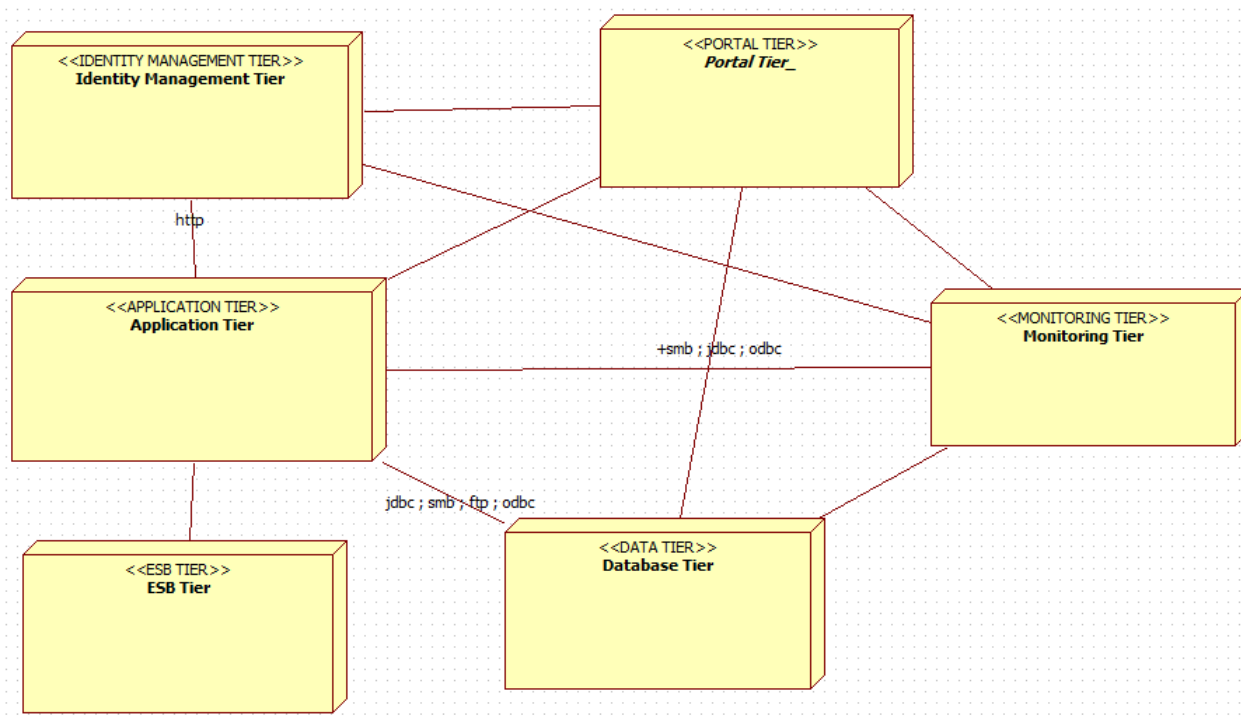
- Open a ssh shell as `opemsa` on main [WLS] (i.e.: `twls10`)
- Change dir to `/var/www/html/emsa_support_files/bcf`
- Run `./csndc-system.sh PROD stop`
- Login as root on [WLS][PMAS][PMAW]
- Change dir to `/var/www/html/emsa_support_files/bcf`
- Edit `./csndc-config.env`
- Run `./csndc-config.sh prod bcf`
- On the `opemsa@twls10` shell, run `./csndc-system.sh PROD start`

6 Deployment View

The following Deployment diagram models the logical view of main CSN-DC artifacts on deployment targets. This diagram is intended to give an overall view of communication paths between nodes and environment needed in each node.

The boxes in light green color are the leverage servers (assets that are not exclusive for the CSN-DC).

6.1 First decomposition: tiers



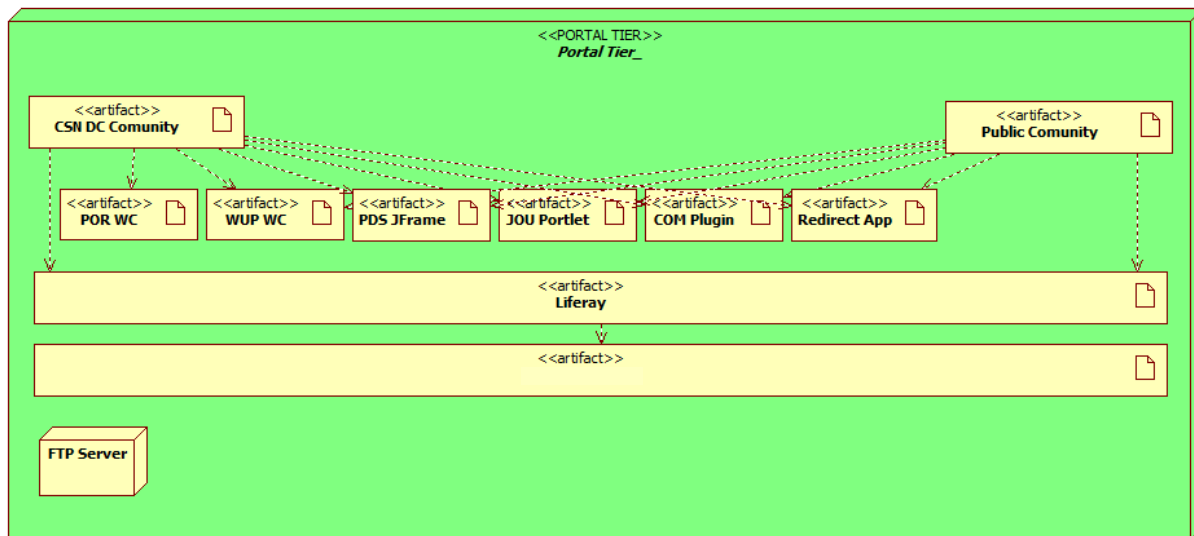
The diagram above describes the CSN-DC Tiers. They are physical implementation of groups of functions and are physically deployed on 1 or many machines. The lines between Tiers indicate the communication channels and where applicable the protocol has been indicated.

NOTE: monitoring tier and ESB Tier, although theoretically available in the CSN-DC environment are not used by the application.

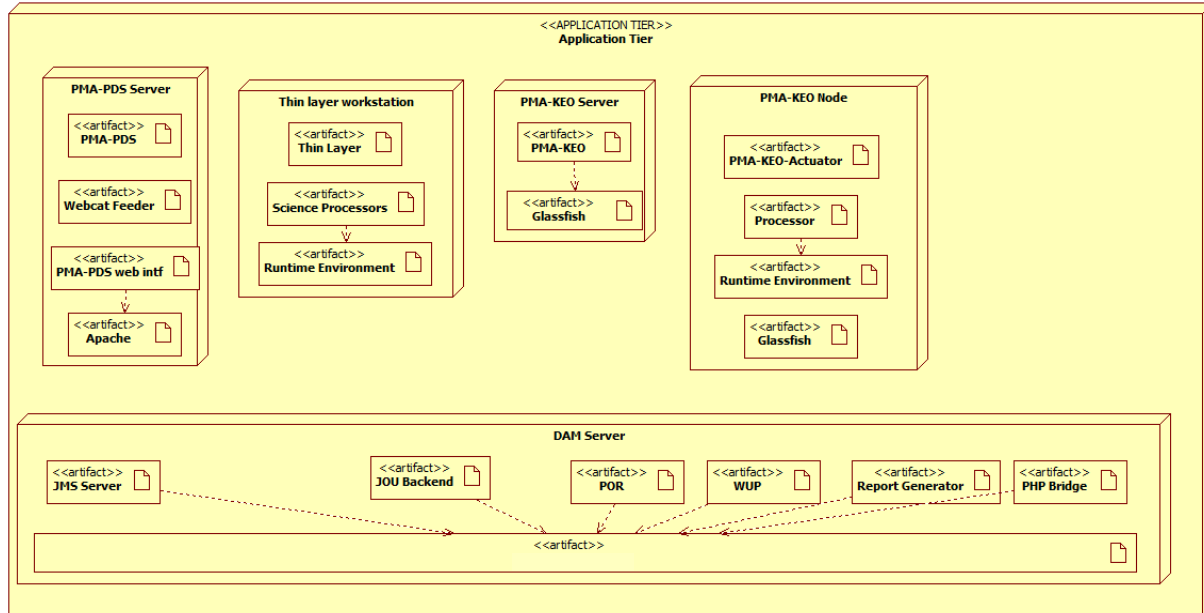
6.2 Portal Tier

The Portal Tier contain all the application exposed on the Liferay Portal. At the lowest level there is the WebLogic. Liferay is deployed on top of WebLogic to which is related with an “uses” relationship

(indicated by the arrow). On top of Liferay the various following *artifacts* are deployed, which implement the functions accessible on the web.



6.3 Application Tier

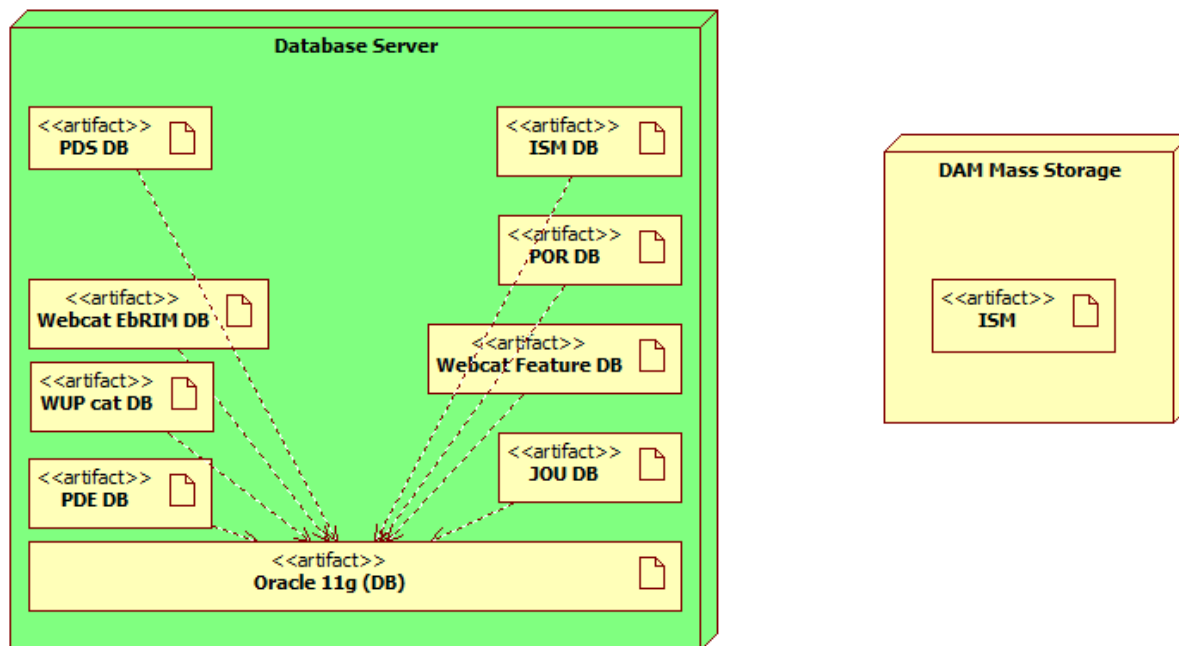


The application tier described above is composed of the following main nodes:

- PMA-PDS Server
- Thin layer workstation
- PMA-KEO server
- PMA-KEO node
- DAM Server

Runtime environment in PMA-KEO node and Thin Layer Workstation are the environment needed by the data manipulation processors (e.g. Envi based applications, etc.).

6.4 Database Server Tier

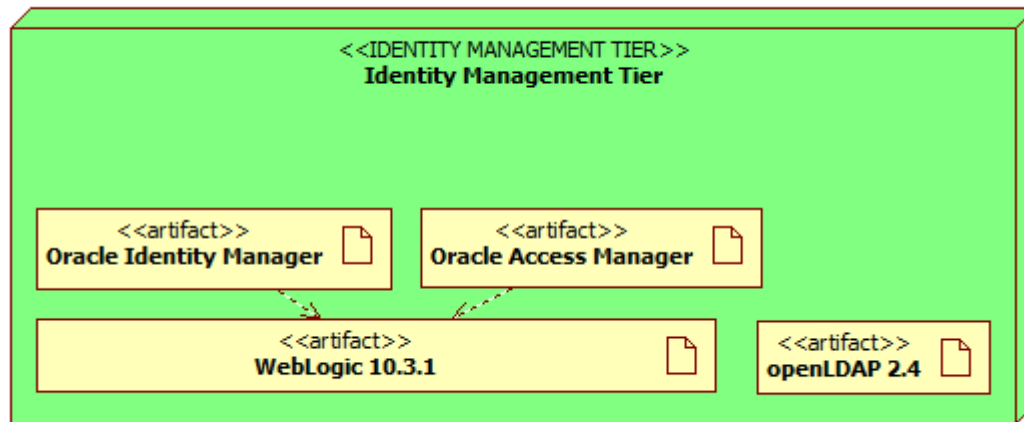


The data Tier is illustrated in the figure above. It is composed by 2 major nodes:

- Database server, based on leverage infrastructure
- DAM Mass Storage, based on the ISM COTS

On the database server there is a number of local application databases (e.g. the PDS DB, POR DB, etc.) which are implemented over the Oracle 11g DBMS.

6.5 Identity management TIER



The Identify Management Tier (see above) is based on 3 major products:

- OpenLADP for storing directory information
- Oracle Identify Manager, for user management (e.g. provisioning, deprovisioning, etc.)
- Oracle Access Manager, for implementing user credential management, e.g. authentication/authorisation, etc.

7 Implementation view

Deprecated, as superseded by [INS].

8 Data View

8.1 DAM

In this section the main data view of the system is exposed. With reference to Section 7 of Functional Design Document, we will expose here how the main data blocks of the CSN-DC system are structured in details. Taking into account the goals of the CSN-DC, the main data blocks defined and stored internally in CSN-DC are:

- Earth Observation Product (EOP) metadata
- Oil Spill feature metadata
- Detected Ship feature metadata

8.1.1 EOP metadata

As per reqs 5.1.0.1-9, of the ITT, the proposed catalogue service for EO Product metadata search is OGC CSW-ebRIM compliant and the corresponding data block is stored in an ebRIM-like database schema. In fact, for EO products there is a clear and quite mature OGC “best practice” for metadata organization in an ebRIM structure.

Following figure reports the Entity Relational (ER) diagram for the EOP metadata data block.

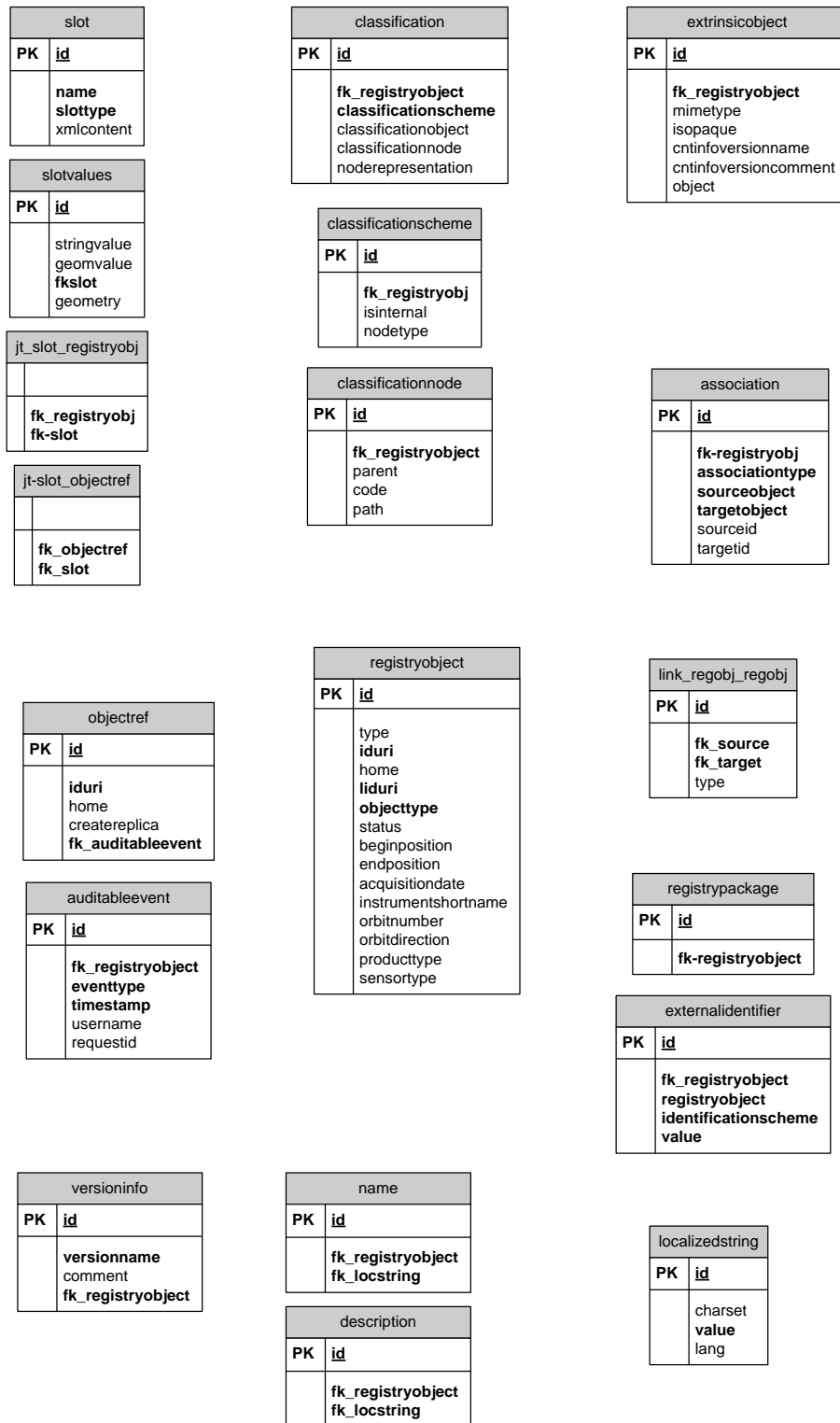


Figure 8-1 ER diagram for EOP metadata

This design allows for a straightforward implementation of the OGC CSW-ebRIM profile for EO Products. In an ebRIM Catalogue, each EO Product instance is represented by an *ExtrinsicObject* (a

special case of a *RegistryObject*) having the *objectType* attribute equal to “urn:x-ogc:specification:csw-ebriim:ObjectType:EO:EOProduct”. This *ExtrinsicObject* is the main object of the EO Product mapping schema. It contains a set of attributes, matching the queryable metadata. These attributes characterize directly the product acquisition and are stored in *Slot* objects referred to the main *ExtrinsicObject*.

Additional information is linked to the main EO Product *ExtrinsicObject*. These additional metadata are stored into specific *ExtrinsicObjects*, linked to the main one using *Associations*. For example, the Acquisition Platform parameters (i.e., Satellite, Instrument and Sensor) are stored in an *ExtrinsicObject* having the *objectType* attribute set to “urn:x-ogc:specification:csw-ebriim:ObjectType:EO:EOAcquisitionPlatform”.

It is linked to the main *ExtrinsicObject* through an *Association* object, with the *associationType* attribute equals to “urn:x-ogc:specification:csw-ebriim:AssociationType:EO:AcquiredBy”.

As a matter of fact, an acquisition Platform metadata set will be common to multiple acquisitions, defining therefore a n:1 association from the EO Product *ExtrinsicObject* to the matching EO Acquisition Platform *ExtrinsicObject*.

A complete description of how EO product metadata are organized in *ExtrinsicObjects* associated or linked with the main one, which metadata they host and how the main object can be *Classified* using *ClassificationNode* objects does, in facts, represent the “EO product data model for the ebRIM profile” OGC best practice (OGC 06-131).

The structure depicted in the above figure can be easily verified against the EO product ebRIM profile.

Moreover, it is optimized for performances and for integration in the DAM.

8.1.2 Oil Spill feature and Detected Ship feature

Differently from EO products, Oil Spill and Detected Ship features have not been standardized so far. Before presenting an ER diagram, we have to introduce the conceptual data model we aim to implement.

The proposed data structure to model an Oil Spill feature is the one proposed in the following figure.

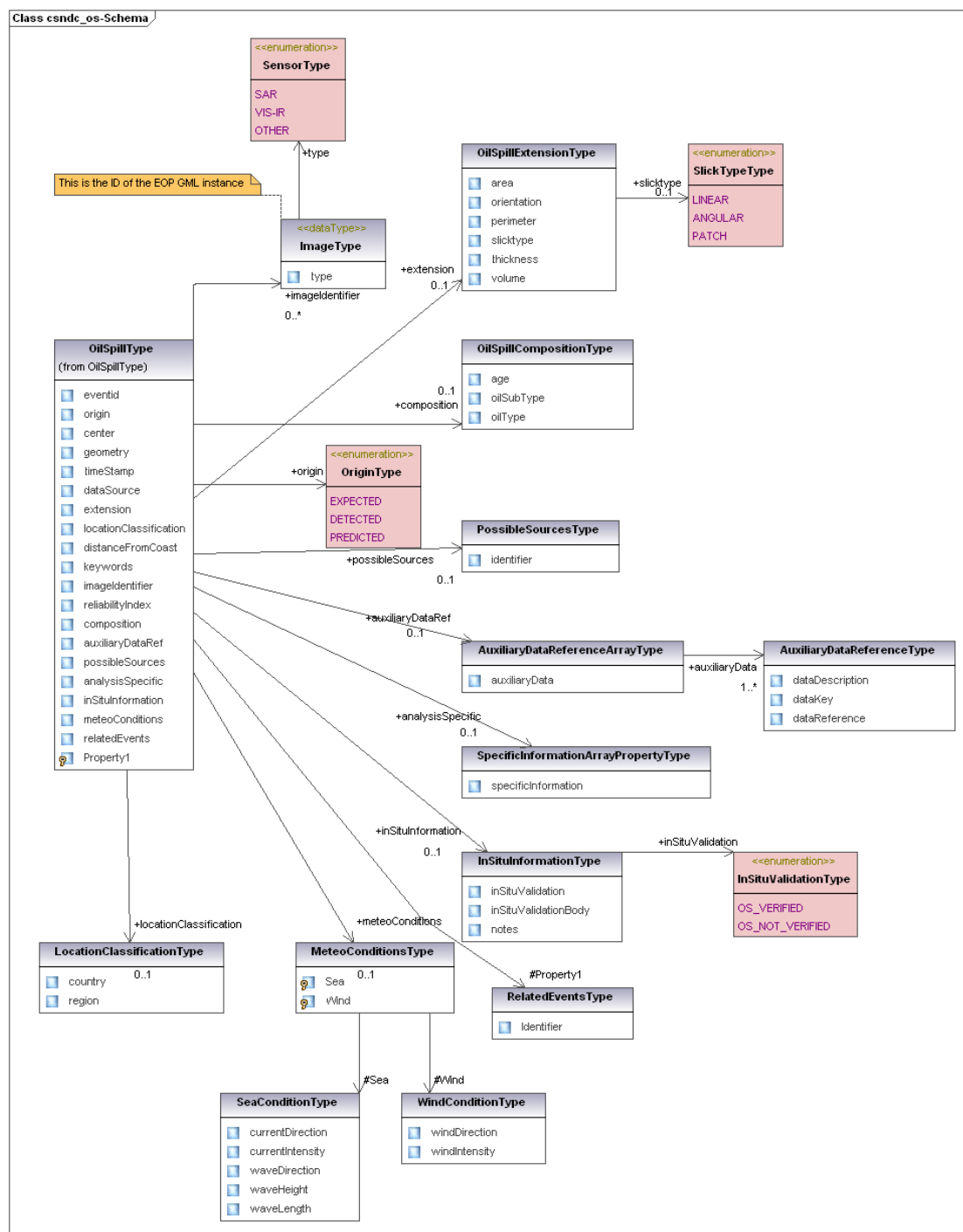


Figure 8-2 Oil Spill feature data model

The description of the classes and attributes in the data model is reported hereafter.

OilSpillType	description	
	Oil Spill feature description	
OilSpillType/eventid	type	string
	description	Unique identifier of the Oil Spill event. For predicted Oil Spills this refers to the observed Oil Spill originating the prediction.
OilSpillType/origin	type	<u>OriginType</u>
	possible values	enumeration EXPECTED enumeration DETECTED enumeration PREDICTED
	description	Observation origin
OilSpillType/center	type	geometric
	description	The center of the Oil Spill expressed as a pos in lat/lon coordinates
OilSpillType/geometry	type	geometric
	description	The polygon describing boundaries of the Oil Spill expressed as one or more Polygon or LinearRing.
OilSpillType/timeStamp	type	dateTime
	description	The date and time of the observation/prediction expressed in ISO8601 format (e.g. '2003-04-01T13:01:02')
OilSpillType/dataSource	type	string
	description	Who provided the observation/prediction
OilSpillType/extension	type	<u>OilSpillExtensionType</u>
	description	Extension parameters of the Oil Spill
OilSpillType/locationClassification	type	<u>LocationClassificationType</u>
	description	Region and country classification
OilSpillType/distanceFromCoast	type	numeric
	description	Approximate distance from nearest coastline.

OilSpillType/keywords	type	string
	description	Valid keywords for free-text search
OilSpillType/imageIdentifier	type	<u>ImageType</u>
	description	The unique identifier of the original EO image(s) used to identify the Oil Spill
OilSpillType/classificationLevel	type	numeric
	description	The probability/confidence that this is a real Oil Spill. Expressed as a number in the [0,1] range. Please note: this will be remapped to class A/class B at application level.
OilSpillType/composition	type	<u>OilSpillCompositionType</u>
	description	Oil type and age information
OilSpillType/auxiliaryDataRef	type	<u>AuxiliaryDataReferenceArrayType</u>
	description	References (as URI) to auxiliary data , images, etc
OilSpillType/possibleSources	type	<u>PossibleSourcesType</u>
	description	Possible sources for the OS
OilSpillType/analysisSpecific	type	<u>SlickTechParametersType</u>
OilSpillType/inSituInformation	type	<u>InSituInformationType</u>
	description	In Situ validation information
OilSpillType/meteoConditions	type	<u>MeteoConditionsType</u>
	description	Sea and wind information associated to the oil spill
OilSpillType/relatedEvents	type	<u>RelatedEventsType</u>
	description	List of other events (i.e. oil spill) that can be put in relation with this Oil Spill (e.g. because of recurrent sources)

AreaType	type	numeric
	description	Value of Oil Spill spatial area quantity.
AuxiliaryDataReferenceArrayType	description	Array of auxiliary data related to the Oil Spill (for example images)
	type	<u>AuxiliaryDataReferenceType</u>
AuxiliaryDataReferenceArrayType/auxiliaryData		
AuxiliaryDataReferenceType	description	Auxiliary data related to the Oil Spill. E.g. the link to an image or other file
AuxiliaryDataReferenceType/dataKey	type	string
	description	Key/identifier of a specific auxiliary data
AuxiliaryDataReferenceType/dataReference	type	string
	description	URI references to auxiliary data
AuxiliaryDataReferenceType/dataDescription	type	string
	description	Description of auxiliary data
ImageType	type	string
	description	EO Image identifier in which the spill is observed
ImageType/type	type	<u>SensorType</u>
	possible values	enumeration SAR enumeration VIS-IR enumeration OTHER
InSituInformationType		
InSituInformationType/inSituValidation	type	<u>InSituValidationType</u>
	possible values	enumeration OS_VERIFIED enumeration OS_NOT_VERIFIED
	description	In Situ validation specifying if Oil Spill presence has been verified
InSituInformationType/inSituValidationBody	type	string
	description	In Situ validation body: who actually verified the OS presence

InSituInformationType/notes	type	string
	description	Free text for notes and observations
LengthType	type	numeric
	description	Linear length of spill's perimeter.
LocationClassificationType	description	Classification of OS location in terms of Region and country
LocationClassificationType/country	type	string
	description	Country
MeteoConditionsType		
MeteoConditionsType/wind	type	<u>WindConditionType</u>
MeteoConditionsType/sea	type	<u>SeaConditionType</u>
OilSpillCompositionType	description	Composition and age parameters associated with the Oil Spill
OilSpillCompositionType/oilType	type	string
	possible values	enumeration Light enumeration Medium enumeration Heavy enumeration OTHER
	description	Composition of the Oil
OilSpillCompositionType/oilSubType	type	string
	description	Sub type of Oil
OilSpillCompositionType/age	type	string
	possible values	enumeration <1 enumeration 1-3 enumeration >3
	description	Age of Oil in days
OilSpillExtensionType	description	Extension and shape parameters associated with the Oil Spill
OilSpillExtensionType/area	type	<u>AreaType</u>
	description	Area of the Oil Spill expressed as Km2.

OilSpillExtensionType/width	type	<u>LengthType</u>
OilSpillExtensionType/length	description	Width of the Oil Spill expressed as m
	type	<u>LengthType</u>
	description	Length of the Oil Spil expressed as m
OilSpillExtensionType/slicktype	type	<u>SlickTypeType</u>
	possible values	enumeration LINEAR enumeration ANGULAR enumeration PATCH
	description	Type of slick (LINEAR, ANGULAR, PATCH)
OilSpillExtensionType/orientation	type	<u>OrientationType</u>
	description	Orientation of the Oil Spil
OilSpillExtensionType/volume	type	string
	possible values	enumeration 0-10 enumeration 10-100 enumeration >100
	description	Volume of the Oil Spill in m3
OilSpillExtensionType/thickness	type	numeric
	description	Thickness of the Oil Spill in mm
OrientationType	type	numeric
	description	Orientation of Oil Spill
PossibleSourcesType	description	Ships, platforms, plants, wrecks that could be the source of the observed spill
PossibleSourcesType/Identifier	type	string
RelatedEventsType	description	Identifier of other events (i.e. oil spills) that can be put in relation with this one
RelatedEventsType /Identifier	type	string
SeaConditionType	description	Sea condition associated to the area of spill

SeaConditionType /dataSource

SeaConditionType /dataType

SeaConditionType/waveHeight

SeaConditionType/waveLength

SeaConditionType/waveDirection

SeaConditionType/currentIntensity

SeaConditionType/currentDirection

SlicktechParametersType

SlicktechParametersType/specificInformation

SlicktechParameterType

SlicktechParameterType/parameter

type	string
type	string
type	numeric
description	Height of the waves expressed in meters
type	numeric
description	Length of the waves expressed in meters
type	numeric
description	Direction of the waves expressed as [0,360] degree value where 0=360=Geographical North, clockwise
type	numeric
description	Intensity of the current expressed in meters/second
type	numeric
description	Direction of the current expressed as [0,360] degree value where 0=360=Geographical North, clockwise
description	Array of ad-hoc analysis metadata.
type	<u>SpecificInformationType</u>
type	string
description	Container for ad-hoc analysis information. The 'specificAttribute' describes the name of the attribute. For example, Shape_characteristics, Contrast_characteristics, Edge_characteristics...

SlicktechParameterType/value

type	string
description	Container for ad-hoc analysys information. The 'specificValue' describes the value of the attribute (see 'specificAttribute').

SlicktechParameterType/description

type	string
description	Container for ad-hoc analysys information. The 'specificDescription' describes in human readable text the meaning and the unit of measure of the attribute (see 'specificAttribute'). This is optional.

WindConditionType

description	Wind condition associated to the area of the spill
-------------	--

WindConditionType /dataSource

type	string
------	---------------

WindConditionType /dataType

type	string
------	---------------

WindConditionType/windIntensity

type	numeric
------	----------------

description	Wind intensity expressed in meters/second
-------------	---

WindConditionType/windDirection

type	numeric
------	----------------

description	Wind direction expressed as [0,360] deegree value where 0=360=Geographical North, clockwise
-------------	---

InSituValidationType

type	string
------	---------------

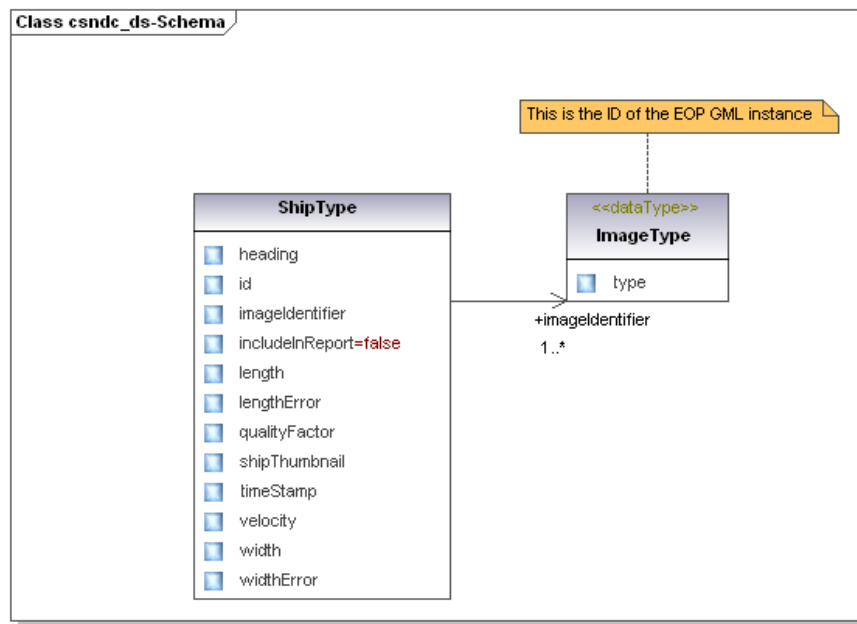
possible values	enumeration OS_VERIFIED enumeration OS_NOT_VERIFIED
-----------------	--

description	In Situ validation specifying if Oil Spill presence has been verified in facts
-------------	--

OriginType	type	string
	possible values	enumeration EXPECTED enumeration DETECTED enumeration PREDICTED
	description	Observation origin of the Oil Spill. It could be EXPECTED meaning that the presence of the Oil Spill is expected as part of a test dataset or insitu independent observation, or DETECTED meaning that the presence of the Oil Spill has been actually detected by the original EO image classification. It is predicted when it comes from a DTOS prediction service.
SensorType	type	string
	possible values	enumeration SAR enumeration VIS-IR enumeration OTHER
	description	EO Sensor Type
SlickTypeType	type	string
	possible values	enumeration LINEAR enumeration ANGULAR enumeration PATCH
	description	Type of slick's shape

Table 8-1 Oil spill feature data model

The above data model can also be expressed as an XML schema that models the Oil Spill feature. This schema will be used as part of the GML application schema for CSN-DC. To have a complete description of the data model, please refer to the GML schema description in Annex A. Following figure reports a data model for Detected Ship features.



Generated by UModel

www.altova.com

Figure 8-3 Detected Ship feature data model

The description of the classes and attributes in the data model is reported hereafter.

ShipType	description	Ship observed in the original satellite image
ShipType/id	type	string
ShipType/includeInReport	type	boolean
	description	If true the ship observation will be included in report
ShipType/timeStamp	type	dateTime
	description	The date and time of the observation expressed in ISO8601 format (e.g. '2003-04-01T13:01:02')
ShipType/heading	type	numeric
	description	Route direction (expressed as as [0,360] deegree value where 0=360=Geographical North)
ShipType/velocity	type	numeric
	description	Velocity (expressed in m/s)
ShipType/length	type	numeric
	description	Length (expressed in meters)
ShipType/lengthError	type	numeric
	description	Error in the estimation of ship length (expressed in meters)
ShipType/width	type	numeric
	description	Width (expressed in meters)
ShipType/widthError	type	numeric
	description	

ShipType/qualityFactor

	Error in the estimation of ship width (expressed in meters)
type	numeric
description	Quality factor (expressed as a percentage)
ShipType/imageIdentifier	
type	ImageType
description	The unique identifier of the original EO image used in which the ship has been detected
ShipType/shipThumbnail	
type	string
description	Name of the thumbnail image file (jpg) with the ship

Table 8-2 Detected ship feature data model

Following figure depicts a database design that matches the proposed Oil Spill and Detected Ships data model. It is optimized for be served through proposed OGC WFS/CSW layer of DAM

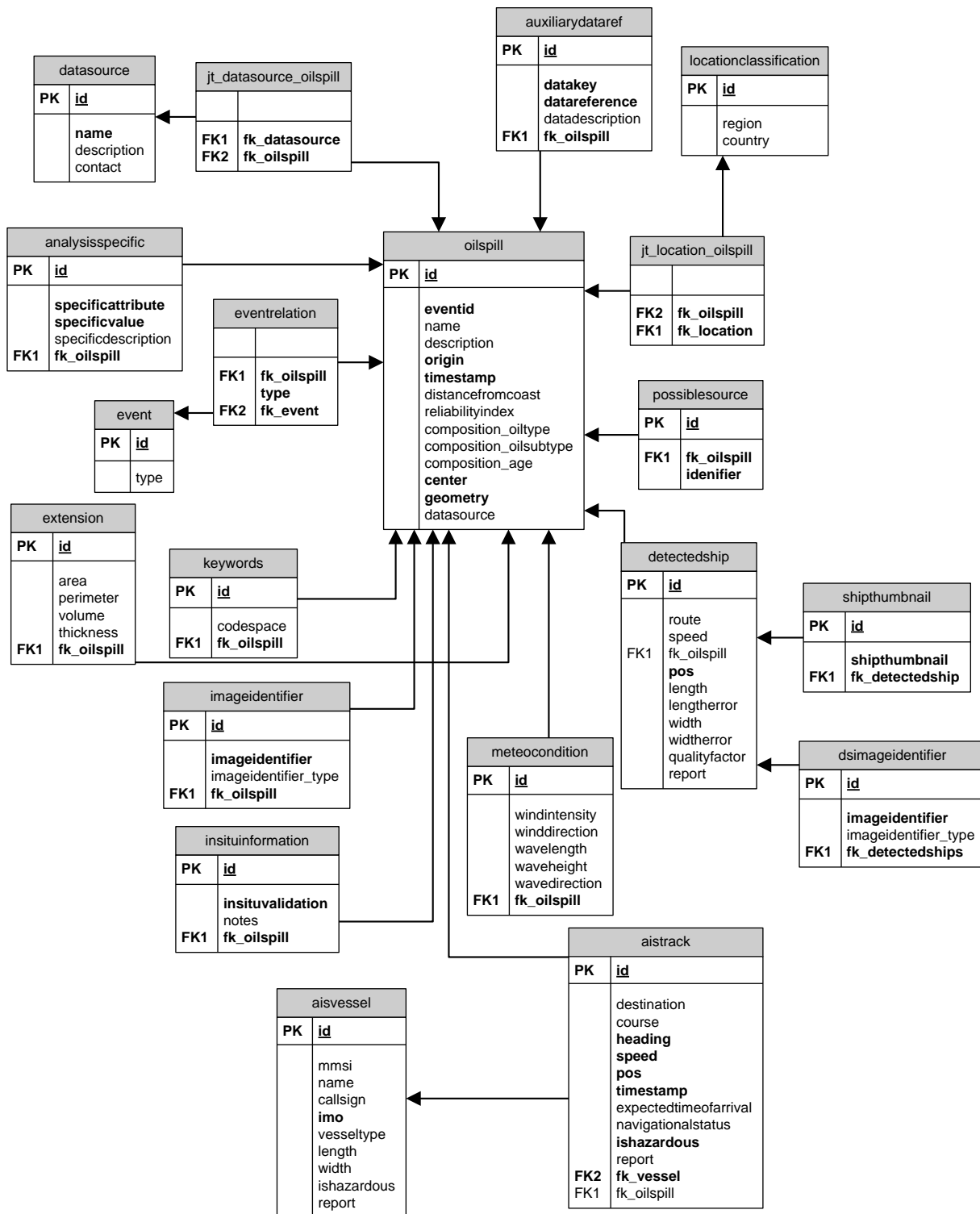


Figure 8-4 ER diagram for detected ship and oil spill metadata

Section 'Data Budget' of Chapter 2 reports sizing estimation of each data type for Webcat DBs. Following table reports a summary of sizing estimation of the Webcat DBs.

Repository	MB/Day	GB/year
Webcat Feature DB (OS and Detected Ships)	145,40	51,83
Webcat Ebrim DB (SAR metadata)	0,07	0,03

See Section 2 for justification of sizing figures.

8.2 POR

The POR ER diagram is reported in the following picture. It is the physical diagram as currently implemented in the POR.

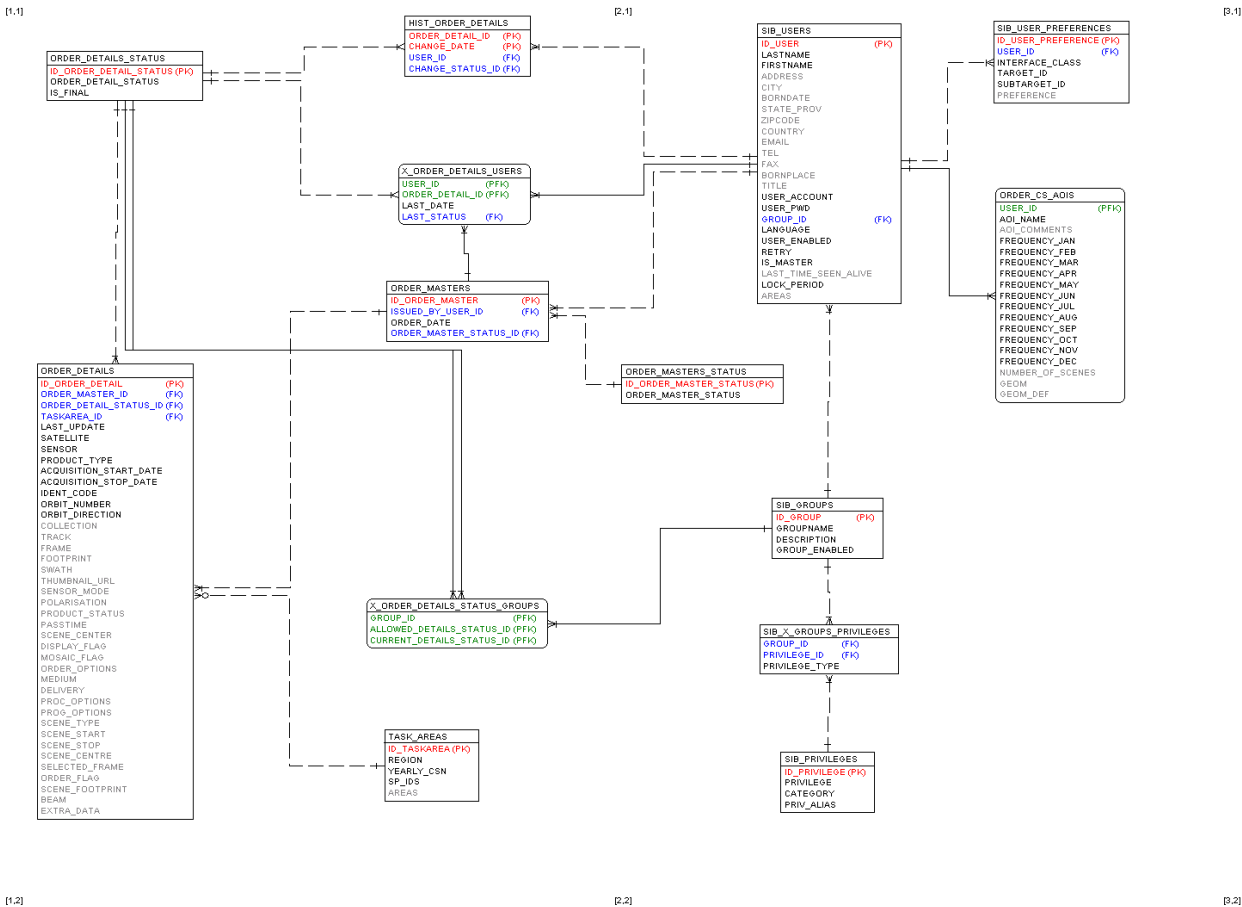


Figure 8-5 ER for the POR component

Section 'Data Budget' of Chapter 2 reports sizing estimation of each data type for POR. Following table reports a summary of sizing estimation of the POR DB.

Repository	MB/Day	GB/year
POR DB	1,20	0,43

See Section 2 for justification of sizing figures.

8.3 COM & JOU

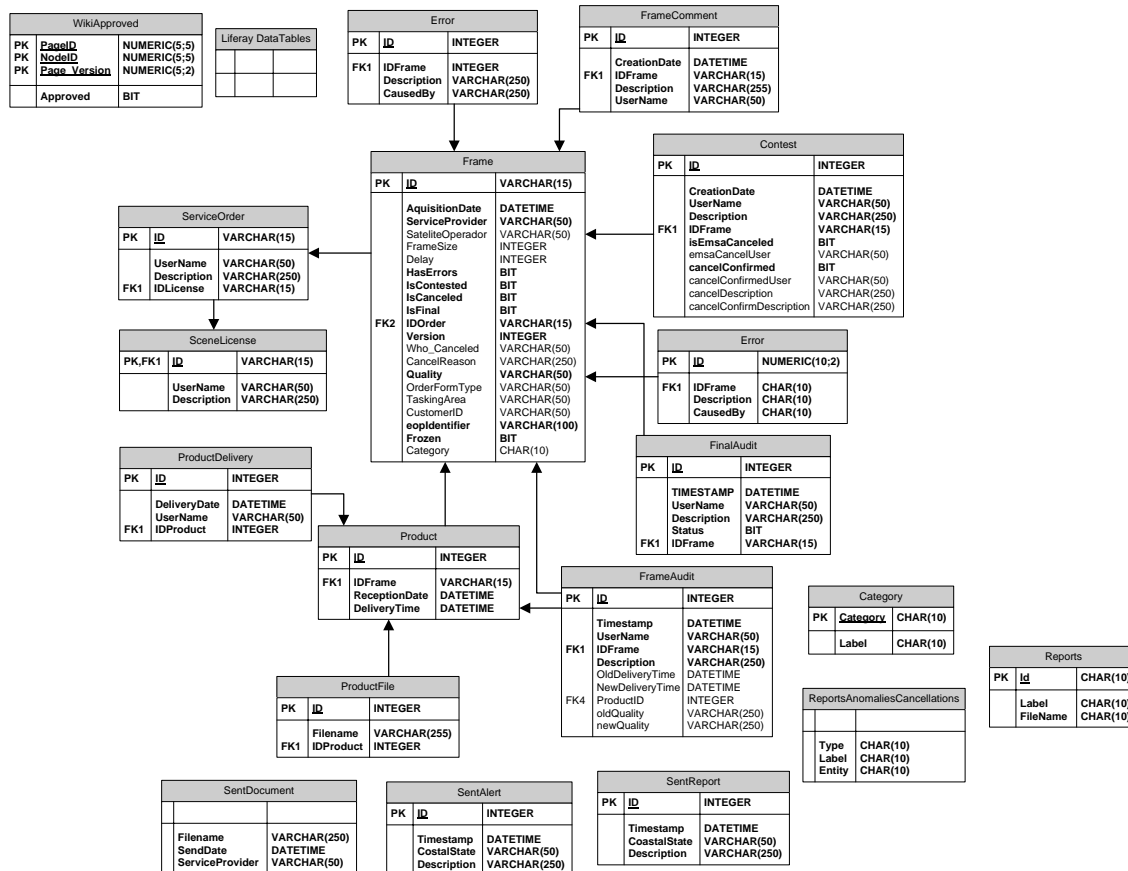


Figure 8-6: COM and JOU data Tables

The figure above represents the COM and JOU data tables. The COM component is almost fully managed by the Liferay internal database, using its tables associated with each of the portlets, represented on the diagram by entity “Liferay DataTables”.

There is one table related to the COM component which was added to improve the Wiki Functionalities: WikiApproved

The journaling component will use the remaining tables:

- **Frame:**
 - Contains all the information regarding frames, that are common to all the products of that frame.
- **FrameComment:**
 - Contains the comments associated to frames, made by the users.
- **Contest:**

- Contains the frame contest information, including cancelations and the reasons for those cancelations.
- **Error:**
 - Contains all information about errors on the frames, and the causes for those errors.
- **ProductFile:**
 - Contains information related to all files associated to the products.
- **Product:**
 - Contains the information about the products associated to frames.
- **ProductDelivery:**
 - Contains all information related to product deliveries.
- **FrameAudit:**
 - Contains audit entries, indicating every change of the reports made.
- **FinalAudit:**
 - Contains information relating to the mark as final such as the user who did it, and the description.
- **ServiceOrder:**
 - Contains the service orders.
- **SceneLicense:**
 - Contains the licences.
- **SentDocument:**
 - Contains a registry of documents sent to SPs.
- **SentAlert:**
 - Contains a registry of alerts sent to coastal states.
- **SentReport:**
 - Contains a registry of reports sent to coastal states.
- **Reports:**
 - Contains the references to the dynamical Jasper-Generated reports

Some tables are used by JOU as temporary tables to facilitate the statistical reports generation, these are:

- **Category**
- **ReportsAnomaliesCancellations**

The data model used by the journaling component will also be used by the JasperReports to provide the reports on the journaled information.

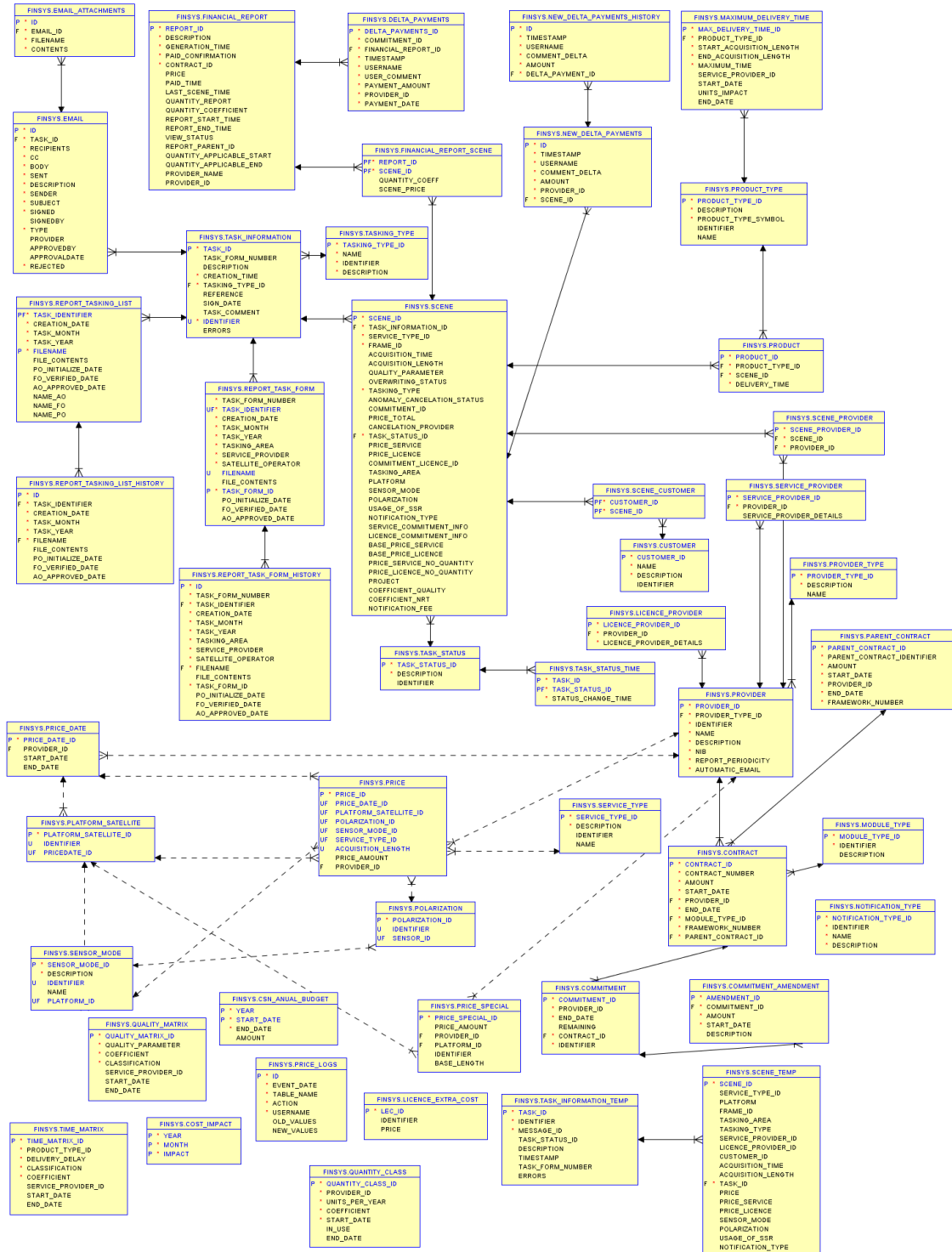
The estimated data growth for COM and JOU together is for 18.7 GB of data per year. This value is detailed in the following tables.

COM Module/Functionality	Growth
Forum data	1 GB
WIKI data	1 GB
Calendar data	2 GB
Documents data	10 GB
TOTAL	14 GB

JOU Module/Functionality	Growth
Service orders	0.1 GB
Scene licenses	0.1 GB
Frames	1.5 GB
Sent documents	0.5 GB
Audit	0.5 GB
Delivered products information	1 GB
Alerts/reports log	1 GB
TOTAL	4.7 GB

8.4 FinSys

Below is a ER model of the Financial System



The main financial system tables are briefly described below:

- **Scene**: contains information regarding the prices and details for a frame
- **Task_Information**: contains information for each cart (task) that was ordered from the POR

- **Product:** contains information relating to the products from the frames that were marked as final on the JOU
- **Delta_Payments:** contains the old delta payments associated with a report, for the current version it is no longer possible to update it via the UI.
- **New_Delta_Payments:** The new delta payments table now associated with a scene and a provider
- **Financial_Report:** contains information about the generated financial reports.
- **Report_Tasking_List:** contains the tasking lists that are generated during the tasking process
- **Report_Task_Form:** contains the tasking forms generated during the tasking process
- **Provider:** contains information about the service and license providers on the financial system
- **Price:** Contains the price configuration, relating to a date; platform, polarization, sensor mode, service type and acquisition length, taken from the respective lookups.
- **Commitment:** The commitment table works as an aggregator for commitment amendments, which contain the real value that is on that commitment. The commitment is related to a provider via a contract.
- **Time_Matrix:** Contains the allowed delay and coefficient for each product per provider so it is considered inside the NRT
- **Quality_Matrix:** Contains the quality coefficient to be applied per provider.
- **Quantity_Class:** contains the default quantity class to be applied per provider.
- **CSN_Annual_Budget:** The budget to be displayed on the graphs, per year
- **Cost_Impact:** The ratio per month to apply the budget

9 External Interfaces

Section removed as the whole information is provided in .

10 Internal Interfaces

The internal interfaces between the CSN-DC components are described in the following table, which includes the following fields:

- ID: unique identifier of the internal interface
- From: direction from which the communication is initiated
- To: direction towards which the communication goes
- Volume: data volume exchanged in Mbytes/day (the volume estimates are based on the same assumptions that are made on the external interfaces definitions)
- Protocol: description of the interface protocol
- Comment: an additional comment, has been included where appropriate

This table includes the interfaces between all CSN-DC components, including also those based on COTS. For example, the interfaces for the PMA-PDS and the PMA-KEO components that are entirely based on pre-existing COTS have also been specified here.

ID	From	To	Data	Format	Volume (MB/day)	Protocol	Comment
IF-IN-01	DAM	PMA-PDS	Data and products	All formats of the products and data ingested (see external interfaces for details)	9000	FTP	It is assumed the worst case scenario in which all data that are ingested are processed in the PMA, e.g. geometric correction.
IF-IN-02	DAM	SYS	Logs and messages	HP OVO Library Format	1	HP OV Agent protocol	
IF-IN-03	DAM	WUP	Products images and metadata	GeoTIFF for images GML for metadata	1000	Virtual File System	The data transfer size depends by the number of users accessing the WUP. By default it has been put equal to the size of SAR data ingested into the system, based on the assumption that all data are visualised on the WUP at least once per day.

IF-IN-04	DAM	PDE	<ul style="list-style-type: none"> • Data notification • Image clips and associated Metadata Maps (wind and wave, situation maps, etc.) 	<ul style="list-style-type: none"> • jpeg/png for image clips and maps • XML for metadata and notification 	150	<ul style="list-style-type: none"> • FTP • REST (4) 	<p>Based on the data notification, the PDE triggers the retrieval of necessary information and rules and creates the alerts and reports.</p> <p>It is assumed that for each processed SAR image, there is an amount of image clips and maps vector layers which amount to 10 MB.</p>
IF-IN-05	DAM	POR	Data notification	XML	1	REST	The DAM notifies the POR about the received data. The POR matches the received SAR data with the order database in order to correctly update the status flag of the received scene.
IF-IN-06	DAM	JOU	Quality Information	XML	1	Web Service	
IF-IN-07	IIF	DAM	Input data and products	All formats of the products and data ingested (see external interfaces for details)	9000	FTP (for the data) Web Services (for commands)	All data arriving to the IIF described in the EIF-04, EIF-7, EIF-08
IF-IN-08	IIF	SYS	Logs and messages	HP OVO Library Format	1	HP OV Agent protocol	
IF-IN-09	IIF	JOU	Notify reception	XML	1	Web Service	Notification of reception of the SP data packages
IF-IN-10	UMA	DAM	Id Mng info	XML		Web Service	
IF-IN-11	UMA	IIF	Id Mng info	XML		Web Service	
IF-IN-12	UMA	SYS	Logs and messages	HP OVO Library Format	1	HP OV Agent protocol	
IF-IN-13	UMA	WUP	Id Mng info	XML		Web Service	
IF-IN-14	UMA	PDE	<ul style="list-style-type: none"> • Id Mng info • report distribution policies • product subscription policies 	XML	0,15	Web service	PDE retrieves the report and product distribution policies in order to be able to manage the report distribution and the subscriptions for delivery of products.
IF-IN-15	UMA	POR	Id Mng info	XML		Web Service	

⁴ The Java Messaging Service (JMS) will be used for transporting these REST messages

IF-IN-16	UMA	JOU	Id Mng info	XML		Web Service	
IF-IN-17	UMA	PMA- KEO	Id Mng info	XML		Web Service	
IF-IN-18	UMA	PMA- PDS	Id Mng info	XML		Web Service	
IF-IN-19	SYS	DAM	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-20	SYS	IIF	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-21	SYS	UMA	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-22	SYS	WUP	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-23	SYS	PDE	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-24	SYS	POR	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-25	SYS	JOU	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-26	SYS	PMA- PDS	Control Commands	Operator Initiated Commands (through OVO library)	0,1	HP OV Agent protocol	
IF-IN-27	WUP	DAM	New products uploaded from the users	GeoTIFF Shapefile	0,1	FTP (for the data) Web Service (for commands)	The user uploading the file shall also fill in a minimal set of metadata which shall accompany the file which gets sent These metadata will also include access rights for the uploaded file. These metadata will be transferred to

							the DAM.
IF-IN-28	WUP	UMA	<ul style="list-style-type: none"> • report distribution policies • product subscription policies 	XML	0	Web service	<p>These policies are defined by the users on the WUP and stored as part of the user configuration data on the UMA LDAP. The volume is negligible, as it is infrequently changed and consists of very little data.</p> <p>The only impact on the LDAP is that obviously the LDAP structure shall be updated in order to accommodate this information.</p>
IF-IN-29	WUP	SYS	Logs and messages	HP OVO Library Format	1	HP OV Agent protocol	
IF-IN-30	WUP	PDE	interactive product/data orders	XML	0,1	Web service	
IF-IN-31	WUP	PDS	Trigger oil spill model	TBD	0,1	TBD	
IF-IN-32	PDE	DAM	reports	<ul style="list-style-type: none"> • PDF + XML metadata • XML for text only alerts 	80	Web service (+ FTP for PDF file)	It is assumed that 15 reports per day are generated with an approximate size of 5 MB, plus some additional metadata.
IF-IN-33	PDE	IIF	delivery directive	XML	1	Web service	
IF-IN-34	PDE	SYS	Logs and messages	HP OVO Library Format	1	HP OV Agent protocol	
IF-IN-35	PDE	WUP	order status	XML	0,1	Web service	This is updated whenever a user interactively requests updated order status info from the WUP.
IF-IN-36	PDE	JOU	<ul style="list-style-type: none"> • billing data (data and products distributed to users) • alarm logs 	XML	1	Web service	
IF-IN-37	POR	SYS	Logs and messages	HP OVO Library Format	1	HP OV Agent protocol	

IF-IN-38	POR	JOU	<ul style="list-style-type: none"> • Log scenes licenses • log service orders 	XML	0,1	Web service	POR wil actively call a web service provided by the JOU for pushing scene licenses details and service orders details
IF-IN-39	PMA-PDS	DAM	Products generated by PMA	GeoTIFF + associated GML metadata	1500	FTP	It is assumed that for each processed SAR image there will be 100 MB including post-processed products and report elements.
IF-IN-40	PMA-PDS	IIF	Order request	GML	0,15	Web service	This is the GML query necessary for the IIF in order to retrieve the STIRES data and make them available for the processing
IF-IN-41	PMA-PDS	SYS	Logs and messages	HP OVO Library Format	2	HP OV Agent protocol	
IF-IN-42	PMA-PDS	WUP	Status information exchanged during the processing on demand	XML	0,15	Web services	

Table 10-1 Description of the internal interfaces between the CSN-DC components

11 Annex -A - Security Concepts

11.1 Clean Sea Net Security Concepts

11.1.1 High level security objectives

The availability of the services provided by EMSA is an important security objective as well as the integrity of the data stored.

Moreover, access to restricted data must be protected from unauthorized accesses. The prevention of the disclosure of restricted information is an important security objective.

According to OWASP security level classification, the targeted level for EMSA CleanSeaNet system is OWASP Level 3 – “Design Verification”.

Level 3 is typically appropriate for applications that handle significant business-to-business transactions, implement business-critical or sensitive functions, or process other sensitive assets, which is the case of EMSA CleanSeaNet. Typical external threats to security can come from:

- viruses and worms,
- opportunists,
- determined skilled and motivated attackers using tools including purpose-built scanning tools.

Nonetheless, the threat from insiders is an important security concern, too. The risk is particularly high in periods preceding the replacement of a contractor by another.. The minimisation of the risk posed by insiders is an important security objective. In this regard, providing non-repudiation is important for operators and administrators.

The scope of Level 3 verification includes all code developed or modified for the application, as well as examining the security of all third party components that provide security functionality for the application. Level 3 ensures that security controls themselves are working correctly, and that they are used everywhere within the application they need to be used to enforce application-specific policies.

As compliance to Level 3 implicitly involves compliance to Level 1 and Level 2 requirements, in the following sections a brief description of all the requirements of these levels and envisaged solution for EMSA CSN system will be addressed. However, before analysing all these specifications it is important to stress that security is to be enforced during the entire life cycle of the system.

11.1.2 CSN Software Development Life Cycle

CSN custom components (such as GISViewer, POR, Alerting, JOU, etc.) are subject to a Software Development Life Cycle (SDLC) based on the suggestions provided by OWASP. More specifically, security is addressed during the architectural design phase, the development phase, the deployment phase and the maintenance-operation phases: This is the best way to avoid late awareness of critical issues.

Each phase has security considerations that ensure a cost-effective and comprehensive security program. When a bug is detected early within the SDLC, it can be addressed more quickly and at a lower impact. A security bug is no different from a functional or performance-based bug in this regard.

To enforce this paradigm, developers are constantly educated with regards to new possible threats and development frameworks and libraries (such as Sibilla and all the PDS support libraries) undergo a security revision process. Enterprise Security API (ESAPI) -like libraries have been developed during the last decade

and are currently used by the development team. Education in security testing also helps developers acquire the appropriate mindset to test an application from an attacker's perspective.

Even if Penetration Testing (covered later in sec. 11.1.3.1) is an important tool to validate the security of a given piece of code, other important practices are always taken into consideration during the SDLC, more specifically Manual Inspections & Reviews, Threat Modelling and Source Code Review.

- **Manual inspections:** human-driven reviews that are continuously performed during the development and when technological design decisions must be taken. While the concept of manual inspections and human reviews is simple, they are considered among the most powerful and effective techniques available, especially when a trust-but-verify model is adopted.
- **Threat modelling:** helps system designers think about the security threats that systems and applications might face. In other terms, it can be seen as risk assessment for applications: Designer must develop mitigation strategies for potential vulnerabilities so to focus the attention on the parts of the system that require it. For each single component, threat models are currently developed by definition and classification of its process, of its assets and understanding of possible vulnerabilities and potential threats. The outcome is a set of design and development recommendations and mitigation strategies.
- **Source code review:** is the process of manually checking application's source code for security issues. Many unintentional but significant security problems are also extremely difficult to discover with other forms of analysis or testing, such as penetration testing, making source code analysis one of the most important tool for technical testing. Examples of issues that are particularly conducive to being found through source code reviews include concurrency problems, flawed business logic, access control problems, and cryptographic weaknesses. These issues often manifest themselves as the most harmful vulnerabilities for an application.
During the Software Development Life Cycle, source code review is currently improved by means of automated Source Scanning applications, such as OWASP's Yasca (see 11.1.3.2).

11.1.3 Manual & Automated Verification

This requirement derives from OWASP Level 1 and 2. The scope of verification includes all the code that is developed or modified for the CSN system.

Automated and manual verification consist of two distinct analyses:

- dynamic analysis, via automated application vulnerability scanning tools and manual penetration testing
- static analysis, via automated source code scanning tools and manual inspection of code

11.1.3.1 Dynamic Analysis

Dynamic analysis consists of using automated tools to access application interfaces, while the application is running, in order to detect vulnerabilities in the application's security controls and verifying all dynamic scan results using either manual application penetration testing or code review.

There are two main types of dynamic analysis:

- Vulnerability Scanning
- Penetration Testing

11.1.3.1.1 Vulnerability Scanning

Vulnerability scanning can be performed by means of a web application security scanner, a program that communicates with a web application through the web front-end in order to identify potential security vulnerabilities in the web application and architectural weaknesses.

Web application scanners can look for a wide variety of vulnerabilities, including:

- Input/Output validation: (Cross-site scripting, SQL Injection, etc.)
- Specific application problems
- Server configuration mistakes/errors/version

Like every testing tools, the web application security scanner is not a perfect tool, it has strength and weaknesses.

Because the tool is implementing a dynamic testing method, it cannot cover 100% of the source code of the application and then, the application itself. Moreover, it is really hard to find logical flaws such as the use of weak cryptographic functions, information leakage, etc.

Finally, the tool cannot implement all variants of attacks for a given vulnerability so it generally has a predefined list of attacks.

Proposed solution for Vulnerability Scanning

The following table defines the tools that are to be adopted for EMSA CSN system.

Involved Components	Solution
All Sibilla based components: <ul style="list-style-type: none"> • WUP Core • POR • ISM Administration GUIs • PDS Administration GUIs 	As all these components have a Flex based client, special Web Scanners must be employed. More specifically, HP SWFScan will be used. SWFScan: <ul style="list-style-type: none"> • Decompiles applications built on the Adobe Flash platform to extract the ActionScript code and statically analyzes it to identify security issues such as information disclosure. • Identifies and reports insecure programming and deployment practices and suggests solutions. For further information see https://h30406.www3.hp.com/campaigns/2009/wwcampaign/1-5TUVE/index.php?key=swf
HTML/Javascript rendering components: <ul style="list-style-type: none"> • JOU • COM 	For these kind of applications, there are a lot of commercial and open source tools to be used for automatic scanning. Possible candidates are: <ul style="list-style-type: none"> • Grendel-Scan: Grendel-Scan is an open-source web application security testing tool. It has automated testing module for detecting common web application vulnerabilities, and features geared at aiding manual penetration tests. The only system requirement is Java 5. Windows, Linux and Macintosh builds are available. Some known features of Grendel-Scan : <ul style="list-style-type: none"> • Internal intercepting / testing proxy • HTTP request fuzzer • Manual requests • Automatic file-not-found profiles • Upstream proxy support • HTTP request & connection throttling • HTML form-based authentication: multiple user

	<p>accounts</p> <ul style="list-style-type: none"> • Granular scan settings • Blocked query parameters • URL white-lists & blacklists • Known session ID names <p>Some known modules of Grendel-scan :</p> <ul style="list-style-type: none"> • SQL injection • Error-based • SQL tautologies - experimental • Miscellaneous tests • CRLF injection • Cross-site request forgery (CSRF) • Directory traversal experimental • Generic fuzzing • Information Leakage • Platform error messages • Robots.txt • Comment lister • Web server configuration • Cross-site tracing (XST) • Proxy detection • Application architecture • Input / output flows • Offline website mirror <ul style="list-style-type: none"> • Nikto: an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 3500 potentially dangerous files/CGIs, versions on over 900 servers, and version specific problems on over 250 servers. Scan items and plug-ins are frequently updated and can be automatically updated.
--	---

Table 11-1 Vulnerability scanning tools

11.1.3.1.2 Penetration Testing

Penetration Testing has been a common technique used to test network security for many years. It is also commonly known as black box testing or ethical hacking. Penetration testing is essentially “*the art of testing a running application remotely, without knowing the inner workings of the application itself, to find security vulnerabilities*”. The tester acts like an attacker and attempts to find and exploit vulnerabilities.

It's worth to stress that, even if penetration testing are useful, they cannot effectively address many of the issues that need to be tested, and in many circumstances it can find flaws in the software that cannot be easily mitigated. Hence, the correct approach is a balanced one that includes several techniques, from manual interviews to technical testing. The balanced approach is sure to cover testing in all phases of the Software Development Life Cycle (see 11.1.2).

What follows is a suggested set of penetration testing according to OWASP specifications.

Suggested Set of Penetration Testing

OWASP Code	Name	Risk	Remarks
OWASP-IG-004	Web Application Fingerprint	Knowing the version and type of a running web/application server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing	Information can be gathered by inspection on the HTTP headers. Applications such as httpprint (http://net-square.com/httpprint/index.shtml) or Netcraft (http://www.netcraft.com) can be used to get these information.
OWASP-IG-005	Application Discovery	A paramount step in testing for web application vulnerabilities is to find out which particular applications are hosted on a web server.	A port scanner and DNS lookup tools can be used to detect unknown services running.
OWASP-IG-006	Analysis of Error Codes	Error codes are very useful to penetration testers during their activities because they reveal a lot of information about databases, bugs, and other technological components directly linked with web applications.	Some information can be gathered by asking for non-existing pages, injecting wrong parameters and performing a wrong login.
OWASP-CM-001	SSL Testing	Some wrong configurations in server installation could be used to force the use of a weaker cipher to gain access to the supposed secure communication channel.	Nessus scanner (http://www.nessus.org) has the capability of checking SSL services on arbitrary ports (generally 443) and report weak ciphers
OWASP-CM-002	ORACLE DB Listener Testing	If an intranet user can access this service he can: <ul style="list-style-type: none"> • Stop the Listener (DoS attack). • Set a password and prevent others from controlling the Listener – (Hijack the DB). • Write trace and log files to any file accessible to the process owner of <i>tnslsnr</i> (Oracle) – (Information leakage). • Obtain detailed information on the Listener, database, and application configuration. 	A special tool develop by Integrigy can help perform the penetration (for more information Oracle Database Listener Security Guide http://www.integrigy.com/securityresources/whitepapers/Integrigy_Oracle_Listener_TNS_Security.pdf)
OWASP-CM-007	Infrastructure and Application Admin Interfaces	Administrator interfaces may be present in the application or on the application server to allow certain users to undertake privileged activities on the site.	Attack techniques include: <ul style="list-style-type: none"> • Directory and file Enumeration • Comments and links in Source • Reviewing Server and Application Documentation

			<ul style="list-style-type: none"> • Alternative Server Port • Parameter Tampering
OWASP-CM-008	Testing for Http methods and Cross Site Tracing (XST)	HTTP offers a number of methods that can be used to perform actions on the web server. Many of these methods are designed to aid developers in deploying and testing HTTP applications. These HTTP methods can be used for nefarious purposes if the web server is not configured properly.	Penetration can be performed via a simple telnet connection and a good knowledge of RFC 2616 specifications. XST attacks can be performed by exploiting the TRACE option of the web server (for more information see Jeremiah Grossman: "Cross Site Tracing (XST)" - http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf)
OWASP-AT-001	Credential Transport over Encrypted Channel	The analysis focuses simply on trying to understand if the data travels unencrypted from the web browser to the server, or if the web application takes the appropriate security measures using a protocol like HTTPS.	Charles, WebScarab or any other proxy tool can be used to perform the test.
OWASP-AT-002	Testing for User Enumeration	Often, web applications reveal when a username exists on system, either as a consequence of a bad configuration or as a design decision. The scope of this test is to verify if it is possible to collect a set of valid usernames by interacting with the authentication mechanism of the application.	Charles, WebScarab or any other proxy tool can be used to perform the test.
OWASP-AT-003	Default or guessable User Accounts	This test aims at finding if default accounts have been unintentionally left on COTS or hardware appliances.	<p>Knowledge of the applications running on the attacked servers is fundamental. Important information can be found at Government Security - Default Logins and Passwords for Networked Devices http://www.governmentsecurity.org/articles/DefaultLoginsandPasswordsforNetworkedDevices.php.</p> <p>Typical tools used for this kind of attack include:</p> <ul style="list-style-type: none"> • Burp Intruder (http://portswigger.net/intruder) • THC Hydra (http://www.thc.org/thc-hydra) • Brutus (http://www.hoobie.net/brutus)
OWASP-AT-005	Testing for Bypassing	Understatement of security threats often result in authentication schemes that can be bypassed	There are several methods to bypass the authentication schema in use by a web application:

	Authentication Schema	by simply skipping the login page and directly calling an internal page that is supposed to be accessed only after authentication has been performed.	<ul style="list-style-type: none"> • Direct page request (forced browsing) • Parameter Modification • Session ID Prediction • SQL Injection
OWASP-AT-006	Testing for Vulnerable Remember Password and Password Reset	Most web applications allow users to reset their password if they have forgotten it, usually by sending them a password reset email and/or by asking them to answer one or more "security questions". This test checks that this function is properly implemented and that it does not introduce any flaw in the authentication scheme. Moreover, it is checked whether the application allows the user to store the password in the browser ("remember password" function).	The key to successfully exploiting and bypassing a password self-reset is to find a question or set of questions which give the possibility of easily acquiring the answers.
OWASP-AT-007	Testing for Logout and Browser Cache Management	This is to check that the logout function is properly implemented, and that it is not possible to "reuse" a session after logout. Moreover, it checks that the application automatically logs out a user when that user has been idle for a certain amount of time, and that no sensitive data remains stored in the browser cache.	Browser back button and cookie inspection is a simple yet effective method of penetration.
OWASP-SM-001	Testing for Session Management Schema	Session Management can be exploited by a penetration tester to gain access to a user account, without the need to provide correct credentials.	<p>It is important to check that cookies and other session tokens are created in a secure and unpredictable way.</p> <p>OWASP's WebScarab features a session token analysis mechanism.</p>
OWASP-SM_003	Testing for Session Fixation	When an application does not renew the cookie after a successful user authentication, it could be possible to find a session fixation vulnerability and force a user to utilize a cookie known by the attacker. In that case an attacker could steal the user session (session hijacking).	OWASP's WebScarab features a session token analysis mechanism.
OWASP-SM-004	Testing for Exposed Session Variables	The Session Tokens (Cookie, SessionID, Hidden Field), if exposed, will usually enable an attacker to impersonate a victim and access the application illegitimately. As such, it is important	OWASP Testing Guide gives detailed information on how to perform this attack.

		that it is protected from eavesdropping at all times – particularly whilst in transit between the Client browser and the application servers.	
OWASP-AZ-003	Testing for Privilege Escalation	This test is performed to check the issue of vertically escalating privileges from one stage to another. During this phase, the tester should verify that it is not possible for a user to modify his or her privileges/roles inside the application in ways that could allow privilege escalation attacks.	OWASP Testing Guide gives detailed information on how to perform this attack.
OWASP-DV-001	Testing for Reflected Cross Site Scripting	Reflected Cross-site Scripting (XSS) is another name for non-persistent XSS, where the attack doesn't load with the vulnerable web application but is originated by the victim loading the offending URI.	OWASP Testing Guide gives detailed information on how to perform this attack. XSS-Proxy (http://xss-proxy.sourceforge.net) is an advanced Cross-Site-Scripting (XSS) attack tool.
OWASP-DV-002	Testing for Stored Cross Site Scripting	Stored Cross Site Scripting (XSS) is the most dangerous type of Cross Site Scripting. Web applications that allow users to store data are potentially exposed to this type of attack. This chapter illustrates examples of stored cross site scripting injection and related exploitation scenarios.	This attack is not so dangerous for Flex applications such as WUP core and POR as javascript in this context is harmless. In any case, other components may be affected. Suggested tools for the attack are <ul style="list-style-type: none"> • BeEF (http://www.bindshell.net/tools/beef) • XSS-Proxy (http://xss-proxy.sourceforge.net) • Backframe (http://www.gnucitizen.org/projects/backframe)
OWASP-DV004	Testing for Cross Site Flashing	ActionScript, like every other language, has some implementation patterns which could lead to security issues. In particular, since Flash/Flex applications (such as WUP core and POR) are embedded in browsers, vulnerabilities like DOM based Cross Site Scripting could be present in flawed Flash applications.	As new Flash players have enforced security against possible attacks, still decompilation may be a security issue. In fact, as SWF files are interpreted by a virtual machine embedded in the player itself, they can be potentially decompiled (for example with Flare , http://www.nowrap.de/flare.html) and analysed. It's worth to stress that in any case, Flex is currently the safest client side language with respect to Javascript that can be easily read and tampered.
OWASP-DV-005	SQL Injection	A SQL injection attack consists of insertion or	As EMSA Clean Sea Net database is ORACLE, the following

		<p>"injection" of a SQL query via the input data from the client to the application.</p> <p>A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system and, in some cases, issue commands to the operating system.</p>	<p>tools can be used to perform the attack:</p> <ul style="list-style-type: none"> • SQLInjector - http://www.databasesecurity.com/sql-injector.htm • Orascan (Oracle Web Application VA scanner) - http://www.ngssoftware.com/products/internet-security/orascan.php • NGSSquirrelL (Oracle RDBMS VA Scanner) - http://www.ngssoftware.com/products/database-security/ngs-squirrel-oracle.php
OWASP-DV-006	LDAP Injection	<p>LDAP Injection is a server side attack, which could allow sensitive information about users and hosts represented in an LDAP structure to be disclosed, modified or inserted.</p> <p>This is done by manipulating input parameters afterwards passed to internal search, add, and modify functions.</p>	<p>OWASP Testing Guide gives detailed information on how to perform this attack.</p>
OWASP-WS-003	XML Structural Testing	<p>XML needs to be well-formed to function properly. XML which is not well-formed shall fail when parsed by the XML parser on the server side.</p> <p>An XML parser is very CPU labour intensive. Some attack vectors exploit this weakness by sending very large or malformed XML messages.</p>	<p>SoapUI or similar tools can be used o perform this attack.</p> <p>To cope with it, developers must put restrictions on appropriate values in accordance to data validation best practice.</p>
OWASP-WS-007	Replay Testing	<p>A replay attack is a "man-in-the-middle" type of attack where a message is intercepted and replayed by an attacker to impersonate the original sender.</p>	<p>For web services, as with other types of HTTP traffic, a sniffer such as Ethereal or Wireshark can capture traffic posted to a web service and using a tool like WebScarab, a tester can resend a packet to the target server. An attacker can attempt to resend the original message or change the message in order to compromise the host server.</p> <p>Pseudo random Session tokens, Time stamping and SSL prevent unauthorized attempts to replay messages.</p>

Table 11-2 Penetration Testing

11.1.3.2 Static Analysis (Source Code Scanning)

Static analysis consists of using automated tools to search through the application source code to find patterns that represent vulnerabilities. As already anticipated, automated tests alone are not sufficient to verify the correct design, implementation, and use of a security control: a more complex paradigm must be implemented through the entire Software development Life Cycle. However, this is an acceptable verification at Level 1.

The scope of this verification is to:

- perform source code scanning on the application.
- verify all source code scan results using either manual application penetration testing or code review.

Proposed solution for Source Code Scanning

As all components developed for/used by EMSA Clean]SeaNet are written in C, C++, Java and PHP, the **Yasca** Source code Scanning environment is sufficient for the purpose.

Yasca is an OWASP open source program which looks for security vulnerabilities, code-quality, performance, and conformance to best practices in program source code. It leverages external open source programs, such as FindBugs, PMD, JLint, JavaScript Lint, PHPLint, Cppcheck, ClamAV, RATS, and Pixy to scan specific file types, and also contains many custom scanners developed just for Yasca.

It is a PHP command-line tool that generates reports in HTML, CSV, XML, SQLite, and other formats.

Yasca is easily extensible via a plugin-based architecture, so scanning any particular file is as simple as coming up with the rules or integrating external tools.

Yasca also features a simple regular-expression plugin (Grep) that allows new rules to be easily written.

It has few base requirements:

- PHP
- Java 1.5 (for PMD, FindBugs, and Pixy)

and has been tested on Windows XP, Vista, and a few flavours of Linux.

Yasca can be used in a number of different ways, including as a:

- checkpoint within a formal SDLC
- desktop tool for developers
- tool integrated into a source code repository

11.1.4 External Entities to CSN Threats Analysis

EMSA CleanSeaNet system provides many external interfaces with different external entities. A threat analysis is carried out based on the STRIDE methodology, also developed by Microsoft. An overview of STRIDE is presented below. See **[STRIDE]** for a more comprehensive description.

The following table summarises the STRIDE threats:

Threat	ID	Property	Threat Definition	Example(s)
Spoofing	S	Authentication	Impersonating something or someone else	Pretending to be Bill Gates, Microsoft.com or ntdll.dll
Tampering	T	Integrity	Modifying data or code	Modifying a DLL on disk or DCD, or a package as it traverses the LAN
Repudiation	R	Non-repudiation	Claiming to have not performed an action	"I did not send that email", "I did not modify that file", etc.
Information disclosure	I	Confidentiality	Exposing information to someone not authorised to access it	Allowing someone to read the Windows source code, publishing a list of customers on a web site, etc.
Denial of service	D	Availability	Deny or degrade service to users	Crashing windows or a web site, sending a packet and absorbing seconds of CPU time, or routing packets into a black hole
Elevation of privileges	E	Authorisation	Gain capabilities without proper authorisation	Allowing a remote internet user to run commands, elevating privileges from normal user to admin

Table 11-3 STRIDE threats definition

The following table points out the external services provided by EMSA CSN that can be exposed to security threats, grouping the interfaces by their exposed protocol.

Protocol	STRIDE analysis: possible threats	ID	Ext I/F	Internal Module
HTTP	<ul style="list-style-type: none"> Tampering: a man in the middle can modify the transmitted data Repudiation: as there is not authentication enforcement, it's not easy to trace the attacker Denial of Service: external attacker could make a denial of service by issuing many requests at the same time 	EIF-03	STIRES	DAM
		EIF-05	SP	DAM

	As these requests are carried out though the WUP Flex component which is subjected to authentication only a Denial of Service could be performed but the attacker could be easily tracked down, unless he has stolen the credentials.	EIF-12b	SO (Feasibility planning tools)	POR
		EIF-12c	SO (Feasibility planning tools)	POR
HTTPS (SOAP)	As these calls require username and password (in the SOAP header) and as the users are only the Service Providers, an IP based security check could be implemented (for example at the firewall level). In this case no real threat is envisaged.	EIF-06	SP	IIF
SFTP	As these file transfers require username and password and as the users are only a set of well-known entities, an IP based security check could be implemented (for example at the firewall level). In this case no real threat is envisaged.	EIF-04	SP	IIF
		EIF-07	MyOcean	IIF
		EIF-08	EO Data Provider	IIF
		EIF-11	External Processes	PMA
TBD	Independently from the protocol used, a point-to-point rule on the firewall could be sufficient to avoid major threats in this interface.	EIF-10	High Deal	JOU

Table 11-4 CSN-DC External Interfaces threats

Moreover the following threats are associated to the POR workflow. This is not associated to the external interfaces, because part of the workflow is internal to the CSN-DC (via the POR), even if external users access to it. The only interface outside the POR is based on sending e-mails.

Scenario	STRIDE analysis	Example	Comment
Tasking forms sent to the SP and SO (the same applies to the tasking forms sent back countersigned by the SP and SO)	Repudiation	CSN-DC operator denies to have sent the tasking form, or claims that have sent a different one.	
	Tampering	Some actor may maliciously modify the content of the tasking form during their	This is unlikely
	Spoofing	Some external actor, pretending to be the CSN-DC may send <i>fake</i> tasking requests.	This is unlikely

Table 11-5 Threats related to the POR workflow

The above threats will be mitigated using electronic signatures and certified e-mail approach.

11.1.5 About Logging

Logging plays a key role in the detection of attacks, and in the gathering of evidence to identify the source of an attack. This, in turn, can have an important dissuasive effect, in particular towards internal users who might otherwise be tempted to attack the system. When a user knows that he incurs a great risk to be identified, which could lead to very unpleasant consequences for him, he can be greatly discouraged to carry out an attack. Moreover, logging is an important asset of the security of an application architecture, since it can be used to detect flaws in applications (users constantly trying to retrieve a file that does not really exist).

As far as Clean Sea Net system is concerned, logs are expected to be produced by both servers and applications. During a normal operation, log level is set so to take into consideration only Warning, error and Critical messages. If needed, log level can be increased in order to have debug information, normally used by the programmer to analyse a particular bug.

From the security point of view, several issues must be addressed, tested and analysed based on the log contents. As a general rule, as logs may contain sensitive information as well as important clues to find possible attackers, they must be kept for a sufficient time as dictated by the security policy (few months may suffice) but at the same time rotated to avoid useless waste of disk space. Moreover, a backup policy could also be enforced to allow for long-term log analysis.

Access to log files must be restricted and controlled. However, if the server is compromised, its logs can be wiped out by the intruder to clean up all the traces of its attack and methods. If this were to happen the system administrator would have no knowledge of how the attack occurred or where the attack source was located. Actually, most attacker toolkits include a log zapper that is capable of cleaning up any logs that hold given information (like the IP address of the attacker) and are routinely used in attacker's system-level root-kits. However, as all messages are routed to the SYS component, it is sufficient to have a good backup policy of all messages gathered by Open View Operation manager in order to still have good clues on the attacker.

Especially for stateless applications, such as WUP Core and POR, whenever possible, log messages will include the credentials of the user that as performed that particular action, together with a timestamp up to milliseconds. Of source, all servers must have their system time synchronised, so that a particular event can be traced cross-servers.

To avoid disclosure of sensitive information, log messages will not contain risky information, such as user password or other information that could allow malicious attacks even from insiders.

Logs can introduce a Denial of Service condition if they are not properly stored. Obviously, any attacker with sufficient resources could be able to, unless detected and blocked, produce a sufficient number of requests that would fill up the allocated space to log files.

This leads to the necessity of not having log files stored on the partitions of the operating system software or of the applications.

This is not to say that logs should be allowed to grow to fill up the partition they reside in. Growth of server logs must be monitored in order to detect this condition since it may be indicative of an attack.

11.2 Security related to Apache reconfigurations

A number of security issues (listed below) are addressed by a specific apache configuration on the WLS machines.

Issue type	Apache configuration
ISSUE#01 - X-Frame-Options header not set;	Header always append X-Frame-Options SAMEORIGIN
ISSUE#02 - X-Content-Type-Options header not set;	Header set X-Content-Type-Options: nosniff
ISSUE#03 - Incomplete or no cache-control and pragma HTTP header;	Header set Cache-Control "no-cache, no-store, must-revalidate, post-check=0, pre-check=0" Header set Pragma "no-cache" Header set Expires Thu, 19 Nov 1981 08:52:00 GMT
ISSUE#04 - Cookie set without secure flag; ISSUE#05 - Cookie set without HttpOnly flag.	Header edit Set-Cookie ^(.*)\$ \$1;HttpOnly;Secure

11.3 Security related to reflected content and reflected cross-site scripting

Other security issues which are related to the cross site scripting are solved by sanitizing the input (e.g. escaping, removing special characters, etc.), provided by the application to the server for producing the various output formats (e.g. excel, PDF, etc.).

Data sanitization focuses on manipulating the data to make sure it is safe by removing any unwanted bits from the data and normalizing it to the correct form. We expect a plain text string from the grid export.

We will use the php function (<http://php.net/manual/en/function.filter-var.php>) "filter_var" with filter "FILTER_SANITIZE_STRING" to remove any markup tags for the value of the grid cells.

12 Annex B - Alerting Parameters

This sections summarises the alerting parameters and how they are computed inside the CSNDC.

IMPACT

	Relevant for alert level GUI	SP or CSN DC	How to Calculate	Type of Value	Allowed Ranges	Units
Surface Area of detected slick	YES	SP	N/A	Floating Point (3 decimal points)	≥ 0	Km ²
* Distance to sensitive areas	YES	CSN DC	See point ¹ below	Floating Point (1 decimal points)	≥ 0	Km
Distance to shoreline	YES	CSN DC	See point ² below	Floating Point (1 decimal points)	≥ 0	Km

¹ A layer will be provided by EMSA with the sensitive areas polygons. We need to calculate the distance from the spill to the polygon (value > 0) as we do for the shoreline – see point 2 below, or classify the spill inside the polygon (value = 0, or negative?).

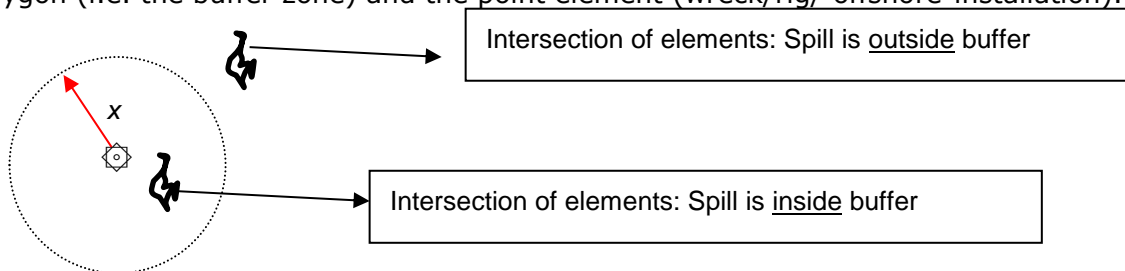
² As discussed in WUP parallel session. Shoreline representation has to be simplified, however ACS needs to provide additional information on the simplified shoreline elements.

CULPRIT

	Relevant for alert level GUI	SP or CSN DC	How to Calculate	Type of Value	Allowed Ranges	Units
Shape of slick aligned with track	YES	SP	N/A	Boolean	YES/NO	N/A
Vessel connected to the detected slick	YES	SP	N/A	Boolean	YES/NO	N/A
Possible polluter identified	YES	SP	N/A	Boolean	YES/NO	N/A
* Distance to TSS or shipping lane	YES	CSN DC	See point ⁴ below	Floating Point (1 decimal points)	≥ 0	Km
* Distance to known wrecks	YES	CSN DC	See point ³ below	Floating Point (1 decimal points)	≥ 0	Km
* Distance to rigs and	YES	CSN DC	See point ³ below	Floating Point	≥ 0	Km

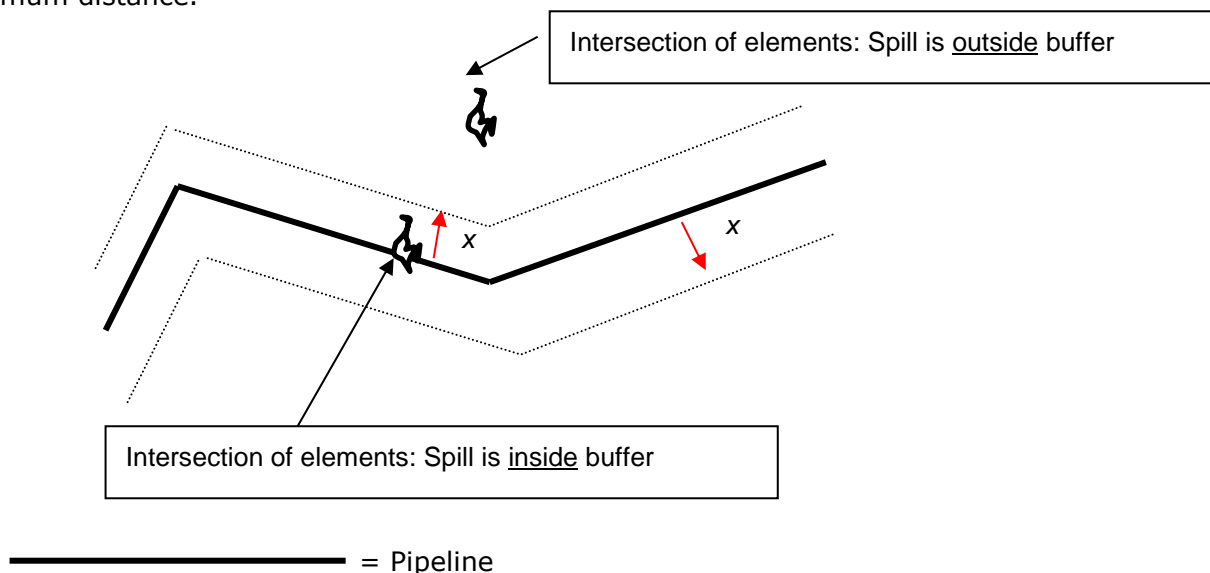
offshore installations				(1 decimal points)		
* Distance to pipeline	YES	CSN DC	See point ⁴ below	Floating Point (1 decimal points)	≥ 0	Km
* Traffic density	YES	CSN DC	See point ⁵ below	Floating Point	≥ 0	Km ⁻²

³ Draw a circular buffer zone around wreck/rig/offshore-installation and calculate if the oil spill falls inside or outside this buffer zone. This can be performed via an Intersection calculation between a polygon (i.e. the buffer zone) and the point element (wreck/rig/ offshore-installation).



⊙ = wreck/rig/ offshore-installation
X = radius of buffer zone

⁴ Draw a buffer zone along the pipeline and calculate if the oil spill falls inside or outside in respect to the minimum distance.



⁵ Traffic density layer will be a raster layer and value will be calculated with mean of pixel data covering the spill polygon.

* EMSA shall provide the layers for the calculation of these parameters

LIKELIHOOD

	Relevant for alert level GUI	SP or CSN DC	How to Calculate	Type of Value	Allowed Ranges	Units
Classification Level (ex-confidence level)	YES	SP	N/A	Booelan	A/B	N/A
Likelihood of a detection being oil	NO	N/A	N/A	N/A	N/A	N/A
Quality of satellite image	NO	SP	N/A	N/A	N/A	N/A
† Algae	NO	SP	N/A	TBC	TBC	TBC
† Ice conditions	NO	SP	N/A	Boolean	0/1	N/A
† Slick already in previous image	NO	SP	N/A	Booelan	YES/NO	N/A
† Daylight condition	NO	CSN DC	Simplified sun ephemerides calculation	% of maximum daylight exposure	$0 \leq x \leq 1$	N/A

MET-OCEAN

	Relevant for alert level GUI	SP or CSN DC	How to Calculate	Type of Value	Allowed Ranges	Units
† Wind Speed	NO	SP	N/A	Floating Point	≥ 0	ms^{-1}
† Wave Height	NO	SP	TBC	TBC	TBC	TBC
† Sea state assessment	NO	SP	TBC	TBC	TBC	TBC
† Currents	NO	SP	TBC	TBC	TBC	TBC

† These parameters are no relevant for the configuration of the alert level, however they will be part of the (PDF) alert report. The CS users may want to configure/customise the alert report by hiding some of these parameters.

13 Annex C - Example of POR Planning files

Here are some examples of the planning files that are managed by the POR.

Here is an example of an Envisat-1 EOLI file.

```
6
Id|Mission|Sensor|Product|COLLECTION|Status|Start|Stop|Orbit|Track|Frame|Swath|PASSTIME|Pass|SCENE_CENTER|FOOT
PRINT|ACQUISITIONDESCRIPTOR|THUMBNAIL_URL|Display|Mosaic|ORDEROPTIONS|Medium|Delivery|Processing
Options|Programming Options|Scene Type|Scene Start|Scene Stop|Scene centre|Selected Frame|Order|SCENE_FOOTPRINT
1|Envisat|ASAR|WS|ASA_WS|ESA.EECF.ENVISAT_ASA_WSx_xS|Potential|2010-07-03 19:52:40.89|2010-07-03
19:54:23.27|43612|472||WS|669492/771876|A|43.50 29.17|40.76 31.85 40.03 26.97 43.00 26.09 46.04 25.10 46.78 30.49 43.73
31.20 40.76 31.85|EN1-043612-669492-771876-WS.XS||true|false|ESA##ESA.acquisition.only##Standard#####Floating
Scene^2010-07-03 19:53:02.15^2010-07-03
19:54:02.50^43.455627^28.615015^860####Polarisation$V/V#####Polarisation: V/V|Floating Scene|2010-07-03
19:53:02.15|2010-07-03 19:54:02.50|+43°27'13" +28°36'55"|860|false|45.562286 30.782804 45.057873 30.900347 44.553337
31.015833 44.047962 31.1315 43.54237 31.245506 43.036743 31.357372 42.530067 31.470123 42.014442 31.582985
41.284813 26.60825 41.800438 26.455454 42.307022 26.302866 42.81187 26.148802 43.316647 25.992115 43.82105
25.832914 44.324028 25.671806 44.826786 25.507914 45.562286 30.782804
2|Envisat|ASAR|WS|ASA_WS|ESA.EECF.ENVISAT_ASA_WSx_xS|Potential|2010-07-07 08:08:51.34|2010-07-07
08:10:31.70|43662|21||WS|2243539/2343896|D|43.45 30.01|46.67 28.68 45.93 34.06 42.73 33.04 40.03 32.22 40.76 27.35
43.47 27.95 46.67 28.68|EN1-043662-2243539-2343896-
WS.XS||true|false|ESA##ESA.acquisition.only##Standard#####Floating Scene^2010-07-07 08:09:10.41^2010-07-07
08:10:10.76^43.43084^30.585642^2735####Polarisation$V/V#####Polarisation: V/V|Floating Scene|2010-07-07
08:09:10.41|2010-07-07 08:10:10.76|+43°25'57" +30°35'09"|2735|false|44.809284 33.689625 44.303482 33.52729 43.797466
33.367718 43.29123 33.210835 42.78636 33.051735 42.280212 32.89845 41.775692 32.742123 41.26215 32.58556 41.99638
27.617304 42.511845 27.731865 43.018356 27.846321 43.524666 27.956446 44.030914 28.072117 44.536983 28.18432
45.042934 28.298481 45.54877 28.41469 44.809284 33.689625
3|Envisat|ASAR|WS|ASA_WS|ESA.EECF.ENVISAT_ASA_WSx_xS|Potential|2010-07-10 08:14:33.56|2010-07-10
08:16:16.61|43705|64||WS|2240845/2343896|D|43.52 28.59|46.83 27.28 46.09 32.67 42.73 31.60 40.03 30.78 40.76 25.91
43.47 26.51 46.83 27.28|EN1-043705-2240845-2343896-
WS.XS||true|false|ESA##ESA.acquisition.only##Standard#####Floating Scene^2010-07-10 08:14:50.29^2010-07-10
08:15:50.64^43.732162^29.228241^2729####Polarisation$V/V#####Polarisation: V/V|Floating Scene|2010-07-10
08:14:50.29|2010-07-10 08:15:50.64|+43°44'02" +29°13'43"|2729|false|45.108307 32.344112 44.603317 32.18093 44.098103
32.02056 43.592667 31.862907 43.087734 31.704502 42.58262 31.54853 42.07726 31.395254 41.563843 31.23723
42.299194 26.244373 42.814587 26.360043 43.320568 26.471067 43.826458 26.584133 44.33222 26.698986 44.83753
26.812372 45.342724 26.927769 45.8478 27.045242 45.108307 32.344112
4|Envisat|ASAR|WS|ASA_WS|ESA.EECF.ENVISAT_ASA_WSx_xS|Potential|2010-07-16 19:44:05.57|2010-07-16
19:45:46.74|43798|157||WS|671535/772712|A|43.58 31.30|40.88 33.98 40.15 29.09 43.59 28.06 46.09 27.23 46.83 32.63 44.32
33.22 40.88 33.98|EN1-043798-671535-772712-WS.XS||true|false|ESA##ESA.acquisition.only##Standard#####Floating
Scene^2010-07-16 19:44:33.17^2010-07-16
19:45:33.52^43.949295^30.635118^870####Polarisation$V/V#####Polarisation: V/V|Floating Scene|2010-07-16
19:44:33.17|2010-07-16 19:45:33.52|+43°56'50" +30°38'08"|870|false|46.055042 32.817936 45.55075 32.93738 45.046333
33.054703 44.54108 33.17227 44.035534 33.288372 43.529957 33.40223 43.02353 33.516514 42.508144 33.63087
41.778656 28.616482 42.29402 28.461512 42.80034 28.306705 43.30502 28.149546 43.809666 27.989729 44.313744
27.827234 44.81622 27.662144 45.318455 27.494146 46.055042 32.817936
5|Envisat|ASAR|WS|ASA_WS|ESA.EECF.ENVISAT_ASA_WSx_xS|Potential|2010-07-19 19:49:48.44|2010-07-19
19:51:31.66|43841|200||WS|669492/772712|A|43.52 29.88|40.76 32.56 40.03 27.68 43.00 26.80 46.09 25.79 46.83 31.19 43.73
31.91 40.76 32.56|EN1-043841-669492-772712-WS.XS||true|false|ESA##ESA.acquisition.only##Standard#####Floating
Scene^2010-07-19 19:50:10.54^2010-07-19
19:51:10.89^43.50305^29.312258^861####Polarisation$V/V#####Polarisation: V/V|Floating Scene|2010-07-19
19:50:10.54|2010-07-19 19:51:10.89|+43°30'04" +29°18'45"|861|false|45.61109 31.48269 45.106182 31.600174 44.60115
31.715595 44.095543 31.831114 43.589706 31.945154 43.08387 32.05679 42.57721 32.169716 42.061592 32.282745
```



```
41.331966 27.30438 41.847572 27.151361 42.354137 26.998545 42.859158 26.843979 43.364178 26.686745 43.868774
26.526987 44.372284 26.365 44.875553 26.200197 45.61109 31.48269
6|Envisat|ASAR|WS|ASA_WS|ESA.EECF.ENVISAT_ASA_WSX_xS|Potential|2010-07-26 08:11:41.98|2010-07-26
08:13:24.16|43934|293||WS|2241717/2343896|D|43.50 29.30|46.77 27.99 46.03 33.37 42.73 32.32 40.03 31.50 40.76 26.63
43.47 27.23 46.77 27.99|EN1-043934-2241717-2343896-
WS.XS||true|false|ESA##ESA.acquisition.only##Standard#####Floating Scene^2010-07-26 08:11:55.33^2010-07-26
08:12:55.68^43.87819^29.987421^2726####Polarisation$V/V#####Polarisation: V/V|Floating Scene|2010-07-26
08:11:55.33|2010-07-26 08:12:55.68|+43°52'46" +29°59'16"|2726|false|45.249767 33.110554 44.74521 32.94664 44.24026
32.78552 43.738033 32.628044 43.232643 32.4723 42.726826 32.312874 42.222733 32.160515 41.709385 32.001762
42.445282 26.996964 42.960648 27.11318 43.465015 27.223679 43.972366 27.340097 44.47756 27.452736 44.9797
27.56667 45.484665 27.683294 45.98934 27.801998 45.249767 33.110554
```

Here is an example of an Radarsat-1 SwathPlanner file (.tbl).

```
"X:\planning-files\2010_07\from osd\RS1_Black_Sea_July_2010_110510.tbl", "07/02/2010 04:17:56.40", "00:45.82", "282D", "Narrow ScanSAR
A", "19.30 -
38.85", "91.6", "48.7", "", "NW:45.12.20N/27.12.59E", "NE:44.40.02N/31.01.15E", "SW:42.30.21N/26.33.45E", "SE:41.58.16N/30.12.00E", "222", "282
.37495", "0.00758", "", "", "", "5", "Realtime"
"X:\planning-files\2010_07\from osd\RS1_Black_Sea_July_2010_110510.tbl", "07/05/2010 15:49:50.78", "00:45.76", "332A", "Narrow ScanSAR
A", "19.30 -
38.85", "100.0", "53.2", "", "NW:44.44.05N/27.06.44E", "NE:45.16.22N/30.55.16E", "SW:42.02.33N/27.55.59E", "SE:42.34.37N/31.34.29E", "222", "33
2.11696", "0.00757", "", "", "", "5", "Realtime"
"X:\planning-files\2010_07\from osd\RS1_Black_Sea_July_2010_110510.tbl", "07/09/2010 04:13:48.32", "00:45.76", "39D", "Narrow ScanSAR
A", "19.30 -
38.85", "96.9", "51.6", "", "NW:44.58.52N/28.12.30E", "NE:44.26.35N/31.59.53E", "SW:42.17.06N/27.33.27E", "SE:41.45.02N/31.10.57E", "223", "39.
37558", "0.00757", "", "", "", "5", "Realtime"
"X:\planning-files\2010_07\from osd\RS1_Black_Sea_July_2010_110510.tbl", "07/26/2010 04:17:54.04", "00:45.82", "282D", "Narrow ScanSAR
A", "19.30 -
38.85", "90.3", "48.0", "", "NW:45.20.39N/27.15.02E", "NE:44.48.21N/31.03.52E", "SW:42.38.41N/26.35.44E", "SE:42.06.36N/30.14.28E", "223", "282
.37456", "0.00758", "", "", "", "5", "Realtime"
"X:\planning-files\2010_07\from osd\RS1_Black_Sea_July_2010_110510.tbl", "07/29/2010 15:49:50.78", "00:45.82", "332A", "Narrow ScanSAR
A", "19.30 -
38.85", "100.0", "53.1", "", "NW:44.44.18N/27.06.40E", "NE:45.16.35N/30.55.13E", "SW:42.02.33N/27.55.59E", "SE:42.34.37N/31.34.29E", "223", "33
2.11696", "0.00758", "", "", "", "5", "Realtime"
```

Here is an example of an Radarsat-2 Acquisition Planning file (.acp).

```
<?xml version="1.0" encoding="UTF-8"?>
<acquisitionCoveragePlan xmlns="http://www.rsi.ca/rs2/ohs/xml/schemas"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.rsi.ca/rs2/ohs/xml/schemas
rs2ohs_acp.xsd">
  <messageType>ACP</messageType>
  <aptConfiguration>Nominal_2010-Feb-24-Version</aptConfiguration>
  <imagingSpecifications>
    <imagingSpecification>
      <satellite>RADARSAT-2</satellite>
      <swathId>swath-1</swathId>
      <beamMnemonic>SCNA</beamMnemonic>
      <transmitPolarization>H</transmitPolarization>
      <receivePolarization>H</receivePolarization>
```

```

<spacecraftOrientation>Right</spacecraftOrientation>
<incidenceAngle>20.86</incidenceAngle>
<imagingStartDate>2010-07-12T18:14:41.430Z</imagingStartDate>
<imagingRelativeOrbitNumber>181.095041892</imagingRelativeOrbitNumber>
<imagingCycle>38</imagingCycle>
<duration>0.007408117</duration>
<conflictFlag>0</conflictFlag>
<downlinkSpecifications>
  <mainPriority>20</mainPriority>
  <subPriority>10</subPriority>
  <receptionFacilityId>TROM</receptionFacilityId>
  <downlinkTimePeriod>
    <downlinkTimeStart>2010-07-12T18:08:41.000Z</downlinkTimeStart>
    <downlinkTimeEnd>2010-07-12T19:15:28.000Z</downlinkTimeEnd>
  </downlinkTimePeriod>
</downlinkSpecifications>
<framingMode>Standard Georeferenced</framingMode>
<numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
<sceneSpecifications>
  <sceneSpecification>
    <sceneNumber>1</sceneNumber>
    <sceneStartTime>2010-07-12T18:14:41.430Z</sceneStartTime>
    <orbitStartTime>13451.095041892</orbitStartTime>
    <sceneDuration>0.007408117</sceneDuration>
    <elevation>222.98281</elevation>
    <sceneArea>28495</sceneArea>
    <percentCoverage>33.4</percentCoverage>
    <sceneCentre>
      <centreLatitude>36.10269</centreLatitude>
      <centreLongitude>-5.23706</centreLongitude>
    </sceneCentre>
    <cornerLists>
      <cornerList>
        <cornerLatitude>37.15646</cornerLatitude>
        <cornerLongitude>-7.13487</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>37.66592</cornerLatitude>
        <cornerLongitude>-3.97650</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>35.02760</cornerLatitude>
        <cornerLongitude>-3.39178</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>34.51822</cornerLatitude>
        <cornerLongitude>-6.44634</cornerLongitude>
      </cornerList>
    </cornerLists>
  </sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
  <arpsSpecification>
    <sceneStartTime>2010-07-12T18:14:41.430Z</sceneStartTime>

```

```

        <orbitStartTime>13451.095041892</orbitStartTime>
        <elevation>199.85240</elevation>
    </arpsSpecification>
    <arpsSpecification>
        <sceneStartTime>2010-07-12T18:15:24.106Z</sceneStartTime>
        <orbitStartTime>13451.102101666</orbitStartTime>
        <elevation>201.23644</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>
    <swathId>swath-2</swathId>
    <beamMnemonic>SCNA</beamMnemonic>
    <transmitPolarization>H</transmitPolarization>
    <receivePolarization>H</receivePolarization>
    <spacecraftOrientation>Right</spacecraftOrientation>
    <incidenceAngle>20.86</incidenceAngle>
    <imagingStartDate>2010-07-13T17:46:19.485Z</imagingStartDate>
    <imagingRelativeOrbitNumber>195.105185705</imagingRelativeOrbitNumber>
    <imagingCycle>38</imagingCycle>
    <duration>0.007419611</duration>
    <conflictFlag>0</conflictFlag>
    <downlinkSpecifications>
        <mainPriority>20</mainPriority>
        <subPriority>10</subPriority>
        <receptionFacilityId>TROM</receptionFacilityId>
        <downlinkTimePeriod>
            <downlinkTimeStart>2010-07-13T17:40:19.000Z</downlinkTimeStart>
            <downlinkTimeEnd>2010-07-13T18:47:06.000Z</downlinkTimeEnd>
        </downlinkTimePeriod>
    </downlinkSpecifications>
    <framingMode>Standard Georeferenced</framingMode>
    <numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
    <sceneSpecifications>
        <sceneSpecification>
            <sceneNumber>2</sceneNumber>
            <sceneStartTime>2010-07-13T17:46:19.485Z</sceneStartTime>
            <orbitStartTime>13465.105185705</orbitStartTime>
            <sceneDuration>0.007419611</sceneDuration>
            <elevation>124.95249</elevation>
            <sceneArea>66250</sceneArea>
            <percentCoverage>77.7</percentCoverage>
            <sceneCentre>
                <centreLatitude>39.71608</centreLatitude>
                <centreLongitude>1.21714</centreLongitude>
            </sceneCentre>
            <cornerLists>
                <cornerList>
                    <cornerLatitude>40.76786</cornerLatitude>
                    <cornerLongitude>-0.77984</cornerLongitude>
                </cornerList>
                <cornerList>
                    <cornerLatitude>41.27884</cornerLatitude>

```

```

        <cornerLongitude>2.54436</cornerLongitude>
    </cornerList>
    <cornerList>
        <cornerLatitude>38.63945</cornerLatitude>
        <cornerLongitude>3.15141</cornerLongitude>
    </cornerList>
    <cornerList>
        <cornerLatitude>38.12977</cornerLatitude>
        <cornerLongitude>-0.04882</cornerLongitude>
    </cornerList>
</cornerLists>
</sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
    <arpsSpecification>
        <sceneStartTime>2010-07-13T17:46:19.485Z</sceneStartTime>
        <orbitStartTime>13465.105185705</orbitStartTime>
        <elevation>109.50499</elevation>
    </arpsSpecification>
    <arpsSpecification>
        <sceneStartTime>2010-07-13T17:47:02.160Z</sceneStartTime>
        <orbitStartTime>13465.112245915</orbitStartTime>
        <elevation>369.80145</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>
    <swathId>swath-3</swathId>
    <beamMnemonic>SCNA</beamMnemonic>
    <transmitPolarization>H</transmitPolarization>
    <receivePolarization>H</receivePolarization>
    <spacecraftOrientation>Right</spacecraftOrientation>
    <incidenceAngle>20.86</incidenceAngle>
    <imagingStartDate>2010-07-16T17:58:11.131Z</imagingStartDate>
    <imagingRelativeOrbitNumber>238.097900444</imagingRelativeOrbitNumber>
    <imagingCycle>38</imagingCycle>
    <duration>0.007428066</duration>
    <conflictFlag>0</conflictFlag>
    <downlinkSpecifications>
        <mainPriority>20</mainPriority>
        <subPriority>10</subPriority>
        <receptionFacilityId>TROM</receptionFacilityId>
        <downlinkTimePeriod>
            <downlinkTimeStart>2010-07-16T17:52:11.000Z</downlinkTimeStart>
            <downlinkTimeEnd>2010-07-16T18:58:58.000Z</downlinkTimeEnd>
        </downlinkTimePeriod>
    </downlinkSpecifications>
    <framingMode>Standard Georeferenced</framingMode>
    <numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
    <sceneSpecifications>
        <sceneSpecification>
            <sceneNumber>3</sceneNumber>
            <sceneStartTime>2010-07-16T17:58:11.131Z</sceneStartTime>

```

```

        <orbitStartTime>13508.097900444</orbitStartTime>
        <sceneDuration>0.007428066</sceneDuration>
        <elevation>247.61366</elevation>
        <sceneArea>27834</sceneArea>
        <percentCoverage>32.6</percentCoverage>
        <sceneCentre>
            <centreLatitude>37.12452</centreLatitude>
            <centreLongitude>-1.28617</centreLongitude>
        </sceneCentre>
        <cornerLists>
            <cornerList>
                <cornerLatitude>38.18083</cornerLatitude>
                <cornerLongitude>-3.21094</cornerLongitude>
            </cornerList>
            <cornerList>
                <cornerLatitude>38.69053</cornerLatitude>
                <cornerLongitude>-0.00847</cornerLongitude>
            </cornerList>
            <cornerList>
                <cornerLatitude>36.04593</cornerLatitude>
                <cornerLongitude>0.58322</cornerLongitude>
            </cornerList>
            <cornerList>
                <cornerLatitude>35.53662</cornerLatitude>
                <cornerLongitude>-2.50987</cornerLongitude>
            </cornerList>
        </cornerLists>
    </sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
    <arpsSpecification>
        <sceneStartTime>2010-07-16T17:58:11.131Z</sceneStartTime>
        <orbitStartTime>13508.097900444</orbitStartTime>
        <elevation>232.38103</elevation>
    </arpsSpecification>
    <arpsSpecification>
        <sceneStartTime>2010-07-16T17:58:53.807Z</sceneStartTime>
        <orbitStartTime>13508.104960219</orbitStartTime>
        <elevation>700.21270</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>
    <swathId>swath-4</swathId>
    <beamMnemonic>SCNA</beamMnemonic>
    <transmitPolarization>H</transmitPolarization>
    <receivePolarization>H</receivePolarization>
    <spacecraftOrientation>Right</spacecraftOrientation>
    <incidenceAngle>20.86</incidenceAngle>
    <imagingStartDate>2010-07-17T17:30:17.360Z</imagingStartDate>
    <imagingRelativeOrbitNumber>252.112705701</imagingRelativeOrbitNumber>
    <imagingCycle>38</imagingCycle>
    <duration>0.007440352</duration>

```

```

<conflictFlag>0</conflictFlag>
<downlinkSpecifications>
  <mainPriority>20</mainPriority>
  <subPriority>10</subPriority>
  <receptionFacilityId>TROM</receptionFacilityId>
  <downlinkTimePeriod>
    <downlinkTimeStart>2010-07-17T17:24:17.000Z</downlinkTimeStart>
    <downlinkTimeEnd>2010-07-17T18:31:04.000Z</downlinkTimeEnd>
  </downlinkTimePeriod>
</downlinkSpecifications>
<framingMode>Standard Georeferenced</framingMode>
<numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
<sceneSpecifications>
  <sceneSpecification>
    <sceneNumber>4</sceneNumber>
    <sceneStartTime>2010-07-17T17:30:17.360Z</sceneStartTime>
    <orbitStartTime>13522.112705701</orbitStartTime>
    <sceneDuration>0.007440352</sceneDuration>
    <elevation>59.23946</elevation>
    <sceneArea>59743</sceneArea>
    <percentCoverage>69.8</percentCoverage>
    <sceneCentre>
      <centreLatitude>42.39344</centreLatitude>
      <centreLongitude>4.72219</centreLongitude>
    </sceneCentre>
    <cornerLists>
      <cornerList>
        <cornerLatitude>43.44498</cornerLatitude>
        <cornerLongitude>2.63768</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>43.95843</cornerLatitude>
        <cornerLongitude>6.10554</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>41.31412</cornerLatitude>
        <cornerLongitude>6.73487</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>40.80312</cornerLatitude>
        <cornerLongitude>3.40891</cornerLongitude>
      </cornerList>
    </cornerLists>
  </sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
  <arpsSpecification>
    <sceneStartTime>2010-07-17T17:30:17.360Z</sceneStartTime>
    <orbitStartTime>13522.112705701</orbitStartTime>
    <elevation>50.71259</elevation>
  </arpsSpecification>
  <arpsSpecification>
    <sceneStartTime>2010-07-17T17:31:00.035Z</sceneStartTime>
    <orbitStartTime>13522.119764766</orbitStartTime>
  </arpsSpecification>

```

```

        <elevation>244.87784</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>
    <swathId>swath-5</swathId>
    <beamMnemonic>SCNA</beamMnemonic>
    <transmitPolarization>H</transmitPolarization>
    <receivePolarization>H</receivePolarization>
    <spacecraftOrientation>Right</spacecraftOrientation>
    <incidenceAngle>20.86</incidenceAngle>
    <imagingStartDate>2010-07-19T18:10:34.422Z</imagingStartDate>
    <imagingRelativeOrbitNumber>281.095850133</imagingRelativeOrbitNumber>
    <imagingCycle>38</imagingCycle>
    <duration>0.007431369</duration>
    <conflictFlag>0</conflictFlag>
    <downlinkSpecifications>
        <mainPriority>20</mainPriority>
        <subPriority>10</subPriority>
        <receptionFacilityId>TROM</receptionFacilityId>
        <downlinkTimePeriod>
            <downlinkTimeStart>2010-07-19T18:04:34.000Z</downlinkTimeStart>
            <downlinkTimeEnd>2010-07-19T19:11:22.000Z</downlinkTimeEnd>
        </downlinkTimePeriod>
    </downlinkSpecifications>
    <framingMode>Standard Georeferenced</framingMode>
    <numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
    <sceneSpecifications>
        <sceneSpecification>
            <sceneNumber>5</sceneNumber>
            <sceneStartTime>2010-07-19T18:10:34.422Z</sceneStartTime>
            <orbitStartTime>13551.095850133</orbitStartTime>
            <sceneDuration>0.007431369</sceneDuration>
            <elevation>320.58616</elevation>
            <sceneArea>25588</sceneArea>
            <percentCoverage>29.9</percentCoverage>
            <sceneCentre>
                <centreLatitude>36.39479</centreLatitude>
                <centreLongitude>-4.25714</centreLongitude>
            </sceneCentre>
            <cornerLists>
                <cornerList>
                    <cornerLatitude>37.45241</cornerLatitude>
                    <cornerLongitude>-6.16336</cornerLongitude>
                </cornerList>
                <cornerList>
                    <cornerLatitude>37.96192</cornerLatitude>
                    <cornerLongitude>-2.99247</cornerLongitude>
                </cornerList>
                <cornerList>
                    <cornerLatitude>35.31556</cornerLatitude>
                    <cornerLongitude>-2.40442</cornerLongitude>
                </cornerList>
            </cornerLists>
        </sceneSpecification>
    </sceneSpecifications>

```

```

        <cornerList>
            <cornerLatitude>34.80621</cornerLatitude>
            <cornerLongitude>-5.46968</cornerLongitude>
        </cornerList>
    </cornerLists>
</sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
    <arpsSpecification>
        <sceneStartTime>2010-07-19T18:10:34.422Z</sceneStartTime>
        <orbitStartTime>13551.095850133</orbitStartTime>
        <elevation>304.24447</elevation>
    </arpsSpecification>
    <arpsSpecification>
        <sceneStartTime>2010-07-19T18:11:17.097Z</sceneStartTime>
        <orbitStartTime>13551.102909907</orbitStartTime>
        <elevation>395.17035</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>
    <swathId>swath-6</swathId>
    <beamMnemonic>SCNA</beamMnemonic>
    <transmitPolarization>H</transmitPolarization>
    <receivePolarization>H</receivePolarization>
    <spacecraftOrientation>Right</spacecraftOrientation>
    <incidenceAngle>20.86</incidenceAngle>
    <imagingStartDate>2010-07-23T17:54:08.633Z</imagingStartDate>
    <imagingRelativeOrbitNumber>338.099454734</imagingRelativeOrbitNumber>
    <imagingCycle>38</imagingCycle>
    <duration>0.007424764</duration>
    <conflictFlag>0</conflictFlag>
    <downlinkSpecifications>
        <mainPriority>20</mainPriority>
        <subPriority>10</subPriority>
        <receptionFacilityId>TROM</receptionFacilityId>
        <downlinkTimePeriod>
            <downlinkTimeStart>2010-07-23T17:48:08.000Z</downlinkTimeStart>
            <downlinkTimeEnd>2010-07-23T18:54:56.000Z</downlinkTimeEnd>
        </downlinkTimePeriod>
    </downlinkSpecifications>
    <framingMode>Standard Georeferenced</framingMode>
    <numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
    <sceneSpecifications>
        <sceneSpecification>
            <sceneNumber>6</sceneNumber>
            <sceneStartTime>2010-07-23T17:54:08.633Z</sceneStartTime>
            <orbitStartTime>13608.099454734</orbitStartTime>
            <sceneDuration>0.007424764</sceneDuration>
            <elevation>163.31541</elevation>
            <sceneArea>38676</sceneArea>
            <percentCoverage>45.3</percentCoverage>
            <sceneCentre>

```



```

        <centreLatitude>37.67745</centreLatitude>
        <centreLongitude>-0.37123</centreLongitude>
    </sceneCentre>
    <cornerLists>
        <cornerList>
            <cornerLatitude>38.73258</cornerLatitude>
            <cornerLongitude>-2.31052</cornerLongitude>
        </cornerList>
        <cornerList>
            <cornerLatitude>39.24247</cornerLatitude>
            <cornerLongitude>0.91665</cornerLongitude>
        </cornerList>
        <cornerList>
            <cornerLatitude>36.59950</cornerLatitude>
            <cornerLongitude>1.51120</cornerLongitude>
        </cornerList>
        <cornerList>
            <cornerLatitude>36.09018</cornerLatitude>
            <cornerLongitude>-1.60365</cornerLongitude>
        </cornerList>
    </cornerLists>
</sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
    <arpsSpecification>
        <sceneStartTime>2010-07-23T17:54:08.633Z</sceneStartTime>
        <orbitStartTime>13608.099454734</orbitStartTime>
        <elevation>153.20347</elevation>
    </arpsSpecification>
    <arpsSpecification>
        <sceneStartTime>2010-07-23T17:54:51.308Z</sceneStartTime>
        <orbitStartTime>13608.106514510</orbitStartTime>
        <elevation>450.99066</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>
    <swathId>swath-7</swathId>
    <beamMnemonic>SCNA</beamMnemonic>
    <transmitPolarization>H</transmitPolarization>
    <receivePolarization>H</receivePolarization>
    <spacecraftOrientation>Right</spacecraftOrientation>
    <incidenceAngle>20.86</incidenceAngle>
    <imagingStartDate>2010-07-27T17:38:17.790Z</imagingStartDate>
    <imagingRelativeOrbitNumber>52.108840521</imagingRelativeOrbitNumber>
    <imagingCycle>39</imagingCycle>
    <duration>0.007419876</duration>
    <conflictFlag>0</conflictFlag>
    <downlinkSpecifications>
        <mainPriority>20</mainPriority>
        <subPriority>10</subPriority>
        <receptionFacilityId>TROM</receptionFacilityId>
        <downlinkTimePeriod>

```

```

        <downlinkTimeStart>2010-07-27T17:32:17.000Z</downlinkTimeStart>
        <downlinkTimeEnd>2010-07-27T18:39:05.000Z</downlinkTimeEnd>
    </downlinkTimePeriod>
</downlinkSpecifications>
<framingMode>Standard Georeferenced</framingMode>
<numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
<sceneSpecifications>
    <sceneSpecification>
        <sceneNumber>7</sceneNumber>
        <sceneStartTime>2010-07-27T17:38:17.790Z</sceneStartTime>
        <orbitStartTime>13665.108840521</orbitStartTime>
        <sceneDuration>0.007419876</sceneDuration>
        <elevation>121.61576</elevation>
        <sceneArea>73669</sceneArea>
        <percentCoverage>86.3</percentCoverage>
        <sceneCentre>
            <centreLatitude>41.01619</centreLatitude>
            <centreLongitude>2.98240</centreLongitude>
        </sceneCentre>
        <cornerLists>
            <cornerList>
                <cornerLatitude>42.06624</cornerLatitude>
                <cornerLongitude>0.94499</cornerLongitude>
            </cornerList>
            <cornerList>
                <cornerLatitude>42.57825</cornerLatitude>
                <cornerLongitude>4.33650</cornerLongitude>
            </cornerList>
            <cornerList>
                <cornerLatitude>39.93993</cornerLatitude>
                <cornerLongitude>4.95296</cornerLongitude>
            </cornerList>
            <cornerList>
                <cornerLatitude>39.42974</cornerLatitude>
                <cornerLongitude>1.69361</cornerLongitude>
            </cornerList>
        </cornerLists>
    </sceneSpecification>
</sceneSpecifications>
<arpsSpecifications>
    <arpsSpecification>
        <sceneStartTime>2010-07-27T17:38:17.790Z</sceneStartTime>
        <orbitStartTime>13665.108840521</orbitStartTime>
        <elevation>98.52819</elevation>
    </arpsSpecification>
    <arpsSpecification>
        <sceneStartTime>2010-07-27T17:39:00.465Z</sceneStartTime>
        <orbitStartTime>13665.115900732</orbitStartTime>
        <elevation>485.04833</elevation>
    </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
<imagingSpecification>
    <satellite>RADARSAT-2</satellite>

```

```

<swathId>swath-8</swathId>
<beamMnemonic>SCNA</beamMnemonic>
<transmitPolarization>H</transmitPolarization>
<receivePolarization>H</receivePolarization>
<spacecraftOrientation>Right</spacecraftOrientation>
<incidenceAngle>20.86</incidenceAngle>
<imagingStartDate>2010-07-30T17:50:18.089Z</imagingStartDate>
<imagingRelativeOrbitNumber>95.102987250</imagingRelativeOrbitNumber>
<imagingCycle>39</imagingCycle>
<duration>0.007410098</duration>
<conflictFlag>0</conflictFlag>
<downlinkSpecifications>
  <mainPriority>20</mainPriority>
  <subPriority>10</subPriority>
  <receptionFacilityId>TROM</receptionFacilityId>
  <downlinkTimePeriod>
    <downlinkTimeStart>2010-07-30T17:44:18.000Z</downlinkTimeStart>
    <downlinkTimeEnd>2010-07-30T18:51:05.000Z</downlinkTimeEnd>
  </downlinkTimePeriod>
</downlinkSpecifications>
<framingMode>Standard Georeferenced</framingMode>
<numberOfUserSpecifiedFrames>0</numberOfUserSpecifiedFrames>
<sceneSpecifications>
  <sceneSpecification>
    <sceneNumber>8</sceneNumber>
    <sceneStartTime>2010-07-30T17:50:18.089Z</sceneStartTime>
    <orbitStartTime>13708.102987250</orbitStartTime>
    <sceneDuration>0.007410098</sceneDuration>
    <elevation>173.22350</elevation>
    <sceneArea>55433</sceneArea>
    <percentCoverage>65.1</percentCoverage>
    <sceneCentre>
      <centreLatitude>38.93204</centreLatitude>
      <centreLongitude>0.36410</centreLongitude>
    </sceneCentre>
    <cornerLists>
      <cornerList>
        <cornerLatitude>39.98312</cornerLatitude>
        <cornerLongitude>-1.60935</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>40.49360</cornerLatitude>
        <cornerLongitude>1.67623</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>37.85692</cornerLatitude>
        <cornerLongitude>2.27730</cornerLongitude>
      </cornerList>
      <cornerList>
        <cornerLatitude>37.34743</cornerLatitude>
        <cornerLongitude>-0.88913</cornerLongitude>
      </cornerList>
    </cornerLists>
  </sceneSpecification>

```

```

</sceneSpecifications>
<arpsSpecifications>
  <arpsSpecification>
    <sceneStartTime>2010-07-30T17:50:18.089Z</sceneStartTime>
    <orbitStartTime>13708.102987250</orbitStartTime>
    <elevation>154.19387</elevation>
  </arpsSpecification>
  <arpsSpecification>
    <sceneStartTime>2010-07-30T17:51:00.765Z</sceneStartTime>
    <orbitStartTime>13708.110047460</orbitStartTime>
    <elevation>461.43532</elevation>
  </arpsSpecification>
</arpsSpecifications>
</imagingSpecification>
</imagingSpecifications>
<regionSpecifications>
  <regionSpecification>
    <regionName>MedWest.rgn</regionName>
    <regionVertexLists>
      <regionVertexList>
        <latitude>41.62970</latitude>
        <longitude>6.33962</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>43.03885</latitude>
        <longitude>6.22592</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>43.28457</latitude>
        <longitude>3.93401</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>39.43243</latitude>
        <longitude>-0.11115</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>38.00577</latitude>
        <longitude>-0.29963</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>36.74838</latitude>
        <longitude>-2.02536</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>36.54628</latitude>
        <longitude>-4.93851</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>37.00877</latitude>
        <longitude>-7.20497</longitude>
      </regionVertexList>
      <regionVertexList>
        <latitude>35.75420</latitude>
        <longitude>-7.17455</longitude>
      </regionVertexList>
    </regionVertexLists>
  </regionSpecification>
</regionSpecifications>

```

```

        </regionVertexList>
        <regionVertexList>
            <latitude>35.86941</latitude>
            <longitude>-2.42550</longitude>
        </regionVertexList>
        <regionVertexList>
            <latitude>38.42002</latitude>
            <longitude>3.04563</longitude>
        </regionVertexList>
        <regionVertexList>
            <latitude>39.67071</latitude>
            <longitude>5.04276</longitude>
        </regionVertexList>
    </regionVertexLists>
</regionSpecification>
</regionSpecifications>
</acquisitionCoveragePlan>

```

Here is an example of an Savoir file (.csv), used for other missions, such as COSMO, TerraSAR-X, etc.

```

Id,Type,Start,End,Duration,Region,OZA,SZA,LookAngle,Min_Incid,Max_Incid,RelOrb,Pass,NW_Lat,NW_Lon,NE_Lat,NE_Lon,SE_Lat,SE_Lon,
SW_Lat,SW_Lon,Center_Lat,Center_Lon,MLST_Start,MLST_End,DataSize,Satellite,Sensor,SensorMode,OrbName,Orbit,Cycle,Track,Frames,
Frame_Start,Frame_End,Revisiting,Slew,Polarisation,Service_Provider
98,ScanSAR (HugeRegion) - Right.FieldOfRegard,27/03/2012 18:15,27/03/2012
18:21,359.0838971,CIRC_N45__W14_K1268,47.40156826,86.28342722,42.0632142,18.9373,60.1444,158.3456517,Descending,56.1865,-
7.8245,55.1998,3.2232,33.3381,-3.4782,34.3677,-10.9073,44.9025,-5.1738,18:13:47,17:57:36,0,Cosmo-SkyMed-1,ScanSAR (HugeRegion) -
Right,Mode3,Sun-Synchronous Nominal Orbit,25990,109,0,13,1,13,,,
99,ScanSAR (HugeRegion) - Right.FieldOfRegard,27/03/2012 19:52,27/03/2012
19:57,282.8069861,CIRC_N45__W14_K1268,32.9868229,88.6441451,29.69953086,18.9461,60.1498,159.3432819,Descending,57.024,-
32.0195,56.034,-20.726,38.8493,-26.3949,39.8535,-34.3929,48.0646,-28.6731,18:14:39,18:00:55,0,Cosmo-SkyMed-1,ScanSAR (HugeRegion)
- Right,FieldOfRegard,Sun-Synchronous Nominal Orbit,25991,109,0,11,1,11,0.067350561,,,EGEOS
100,ScanSAR (HugeRegion) - Right.FieldOfRegard,28/03/2012 06:48,28/03/2012
06:54,369.5167665,CIRC_N45__W14_K1268,46.6616356,89.90450749,41.44746401,18.9303,60.1396,166.0925522,Ascending,56.0109,-
23.56,57.0008,-12.2753,34.55,-9.1145,33.5215,-16.5583,45.4051,-14.9099,05:52:05,05:35:10,0,Cosmo-SkyMed-1,ScanSAR (HugeRegion) -
Right,FieldOfRegard,Sun-Synchronous Nominal Orbit,25998,109,0,14,1,14,0.455646938,,,KSAT
101,ScanSAR (HugeRegion) - Right.FieldOfRegard,28/03/2012 18:33,28/03/2012
18:39,371.7217441,CIRC_N45__W14_K1268,47.28838819,86.08787468,41.96909737,18.9425,60.1472,173.3438714,Descending,56.8157,-
12.2996,55.8266,-1.0685,33.2001,-8.0684,34.2305,-15.4851,45.1521,-9.6978,18:14:26,17:57:31,0,Cosmo-SkyMed-1,ScanSAR (HugeRegion) -
Right,FieldOfRegard,Sun-Synchronous Nominal Orbit,26005,109,0,14,1,14,0.489540538,,,HH_HV_VV_HH,
109,ScanSAR (HugeRegion) - Right.FieldOfRegard,30/03/2012 19:09,30/03/2012
19:15,366.1726166,CIRC_N45__W14_K1268,44.98871049,86.06533628,40.04631894,18.9425,60.1472,203.3416211,Descending,57.6108,-
21.3108,56.6181,-9.8373,34.3435,-16.905,35.3681,-24.4267,46.1217,-18.5944,18:15:17,17:58:11,0,Cosmo-SkyMed-1,ScanSAR (HugeRegion)
- Right,FieldOfRegard,Sun-Synchronous Nominal Orbit,26035,109,0,13,1,13,0.065905871,,,HH_HV,
140,HS Full Performance Right.FieldOfRegard,28/03/2012 07:16,28/03/2012
07:22,356.305911,CIRC_N45__W14_K1268,42.41058877,86.98990191,38.60819884,20.3235,55.4517,155.3414965,Descending,57.4196,-
18.1808,56.6738,-10.4716,34.4241,-17.252,35.1324,-22.3185,46.0011,-17.5872,06:26:53,06:08:15,0,TerraSAR-X,HS Full Performance
Right,FieldOfRegard,Sun-Synchronous Nominal Orbit,26541,159,0,,,,,0.476689674,,,
141,HS Full Performance Right.FieldOfRegard,28/03/2012 17:57,28/03/2012
18:02,304.0144217,CIRC_N45__W14_K1268,38.35630274,84.59651272,35.03632538,20.3105,55.4474,162.1037377,Ascending,56.4761,-
9.1084,57.2205,-1.4387,38.212,2.1256,37.5072,-3.1534,47.4326,-2.4855,17:50:23,17:33:57,0,TerraSAR-X,HS Full Performance
Right,Mode2,Sun-Synchronous Nominal Orbit,26548,159,0,,,,,0.445417088,,,

```


14 Annex D – Example of XML file for triggering the Oil Spill Model WPS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>org.n52.wps.server.algorithm.seatrack.StartManual</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>oilspillPolygon</ows:Identifier>
      <wps:Reference schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"
mimeType="text/xml" xlink:href="http://217.111.153.44:7021/deegree-
wfs/services?SERVICE=WFS&VERSION=1.1.0&REQUEST=GetFeature&featureId=OS_2994"
method="GET"/>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>runId</ows:Identifier>
      <wps>Data>
        <wps:LiteralData dataType="xs:string">217</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>substance</ows:Identifier>
      <wps>Data>
        <wps:LiteralData dataType="xs:string">Medium oils (100-1000
cSt)</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>forwardCalculation</ows:Identifier>
      <wps>Data>
        <wps:LiteralData dataType="xs:boolean">true</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>simulationDuration</ows:Identifier>
      <wps>Data>
        <wps:LiteralData dataType="xs:integer">24</wps:LiteralData>
      </wps>Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>oilType</ows:Identifier>
      <wps>Data>
        <wps:LiteralData dataType="xs:string">Oil classes</wps:LiteralData>
      </wps>Data>
    </wps:Input>
  </wps:DataInputs>
</wps:Execute>
```

```

<wps:Input>
  <ows:Identifier>scenarioName</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:string">Test scenario</wps:LiteralData>
  </wps:Data>
</wps:Input>
<wps:Input>
  <ows:Identifier>scenarioDescription</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:string">A scenario for testing the
wps</wps:LiteralData>
  </wps:Data>
</wps:Input>
<wps:Input>
  <ows:Identifier>outletDepth</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:integer">0</wps:LiteralData>
  </wps:Data>
</wps:Input>
<wps:Input>
  <ows:Identifier>stateOfOil</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:string">FRESH</wps:LiteralData>
  </wps:Data>
</wps:Input>
<wps:Input>
  <ows:Identifier>instantaneous</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:boolean">false</wps:LiteralData>
  </wps:Data>
</wps:Input>
<wps:Input>
  <ows:Identifier>dischargeDuration</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:integer">24</wps:LiteralData>
  </wps:Data>
</wps:Input>
<wps:Input>
  <ows:Identifier>dischargeAmount</ows:Identifier>
  <wps:Data>
    <wps:LiteralData dataType="xs:integer">10</wps:LiteralData>
  </wps:Data>
</wps:Input>
</wps:DataInputs>
<wps:ResponseForm>
  <wps:ResponseDocument storeExecuteResponse="true" lineage="true" status="true">
    <wps:Output asReference="true" mimeType="text/xml" encoding="UTF-8">
      <ows:Identifier>modeloutput</ows:Identifier>
      <ows:Title>modeloutput</ows:Title>
      <ows:Abstract>modeloutput</ows:Abstract>
    </wps:Output>
    <wps:Output asReference="true" mimeType="text/xml" encoding="UTF-8">
      <ows:Identifier>contour</ows:Identifier>
      <ows:Title>contour</ows:Title>
    </wps:Output>
  </wps:ResponseDocument>
</wps:ResponseForm>

```



```
        <ows:Abstract>contour</ows:Abstract>
    </wps:Output>
    <wps:Output asReference="true" mimeType="text/xml" encoding="UTF-8">
        <ows:Identifier>densityNetCDF</ows:Identifier>
        <ows:Title>densityNetCDF</ows:Title>
        <ows:Abstract>densityNetCDF</ows:Abstract>
    </wps:Output>
</wps:ResponseDocument>
</wps:ResponseForm>
</wps:Execute>
```

European Maritime Safety Agency

Praça Europa 4
1249-206 Lisbon, Portugal
Tel +351 21 1209 200
Fax +351 21 1209 210
emsa.europa.eu

