



# ASB CCN#2

## Final Presentation Meeting

4 October 2018

1. Introduction to the ASB project and the CCN2 extension
2. Introduction to the ASB technical concepts
3. Technical achievements and demonstration scenario
4. Live demonstration
5. CCN2 deliverables
6. Concluding remarks
7. Future work

- **Looking for a way to reduce development time and costs for building a processing facility (PF).**
- **ASB established to investigate the potential for automating the building of PFs.**

Key challenges:

- Provide a scalable and dynamically re-configurable platform for processing facilities having variable and complex processing demands such as an Instrument Processing Facilities (IPF).
  - Automate the building of workflows from user specifications making the deployment of the processors in the cloud seamless for users, but under their control.
- 
- **Along the way Principle Investigators and Scientists highlighted the need for support over the development lifecycle as well as on-demand processing in addition to systematic processing.**
    - Need to host on PC, Server, and Cloud
    - Manage and process large data series (Big Data)
    - Complex workflows executable on multiple platforms globally distributed
    - Additional user interfaces

- The ASB implementation is driven by two distinct usages: automated processing in IPFs and support for algorithm development.
- In the main ASB project, two use cases reproducing real-life applications have been prepared:
  - The SMOS L1OP use case reproduces a fragment of the SMOS IPF. This ASB prototype application is processing incoming products without user intervention.
  - The PROBA-V n-Daily Compositor use case reproduces the processing chain of the Compositor implemented in the PROBA-V MEP. The resulting application is meant to be executed on-demand.
- The ASB framework has been selected in project PROBA-V MEP Third Party Services (MEP-TPS) for supporting the development, testing and execution of algorithms developed by scientific partners.
- ASB was lacking functionalities identified as needed in discussions with potential users, and in particular with the MEP-TPS partners.
- **The features selected for implementation in ASB CCN2 partially fill the gap between the main ASB project and the MEP-TPS project.**

## ASB CCN#2 Proposal – ASB-SA-PRO-002-CCN2

### Section 1.4.2 – Relationship CCN2, MEP-TPS Project and Follow-on Work

- Complete
- Work in progress
- To be worked on

#### 1. ASB CCN#2

1a. Addition of the developer role  
(= "contributing users" in the MEP-TPS)

1b. Remote execution of algorithms

1c. User selected processor execution

#### 2. MEP-TPS

2a. Processor execution scheduler module

2b. Remote access to generated products

2c. Programmatic processor execution

#### 3. Follow-on work

3a. Upgrade of the developer's interface

3b. Processor monitoring (enh.)

3c. Knowledge elements versioning (enh.)

3d. Automated build and user interface

3e. Parallelising Large Collections of Data

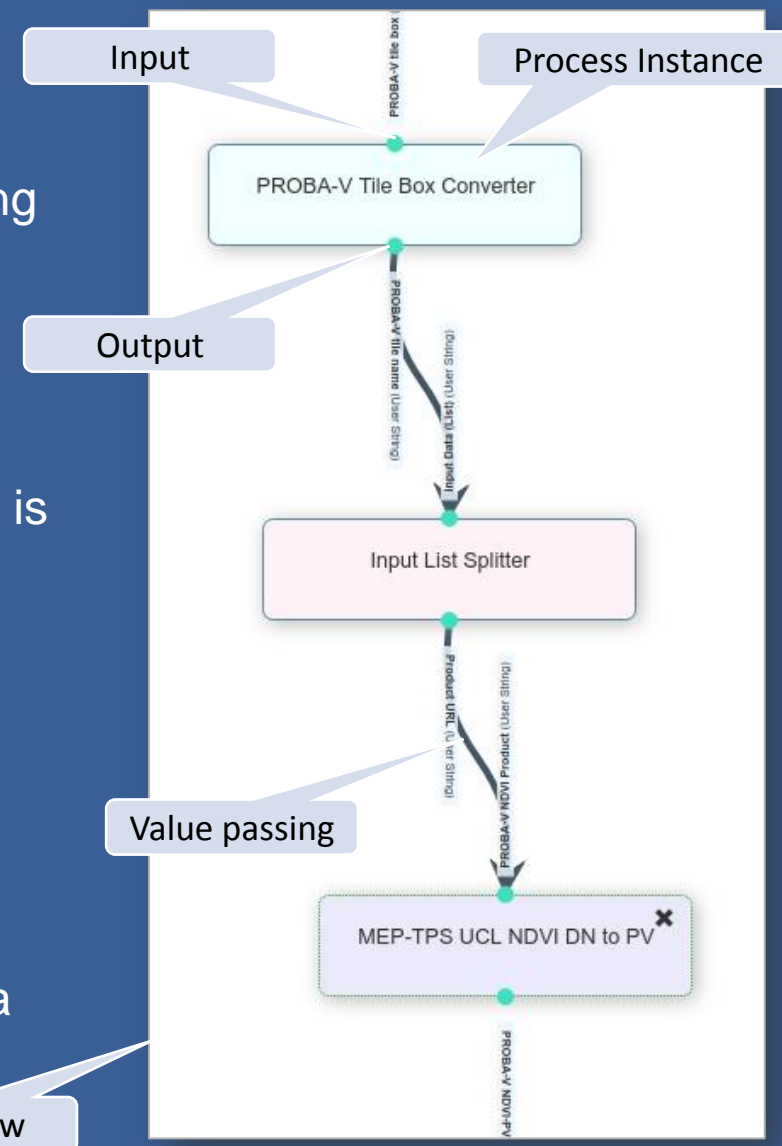
3f. Automate IT resources dynamicity

3g. Data Manager

3h. Processor execution reporting (enh.)

3i. Extreme Real-life conditions and testing

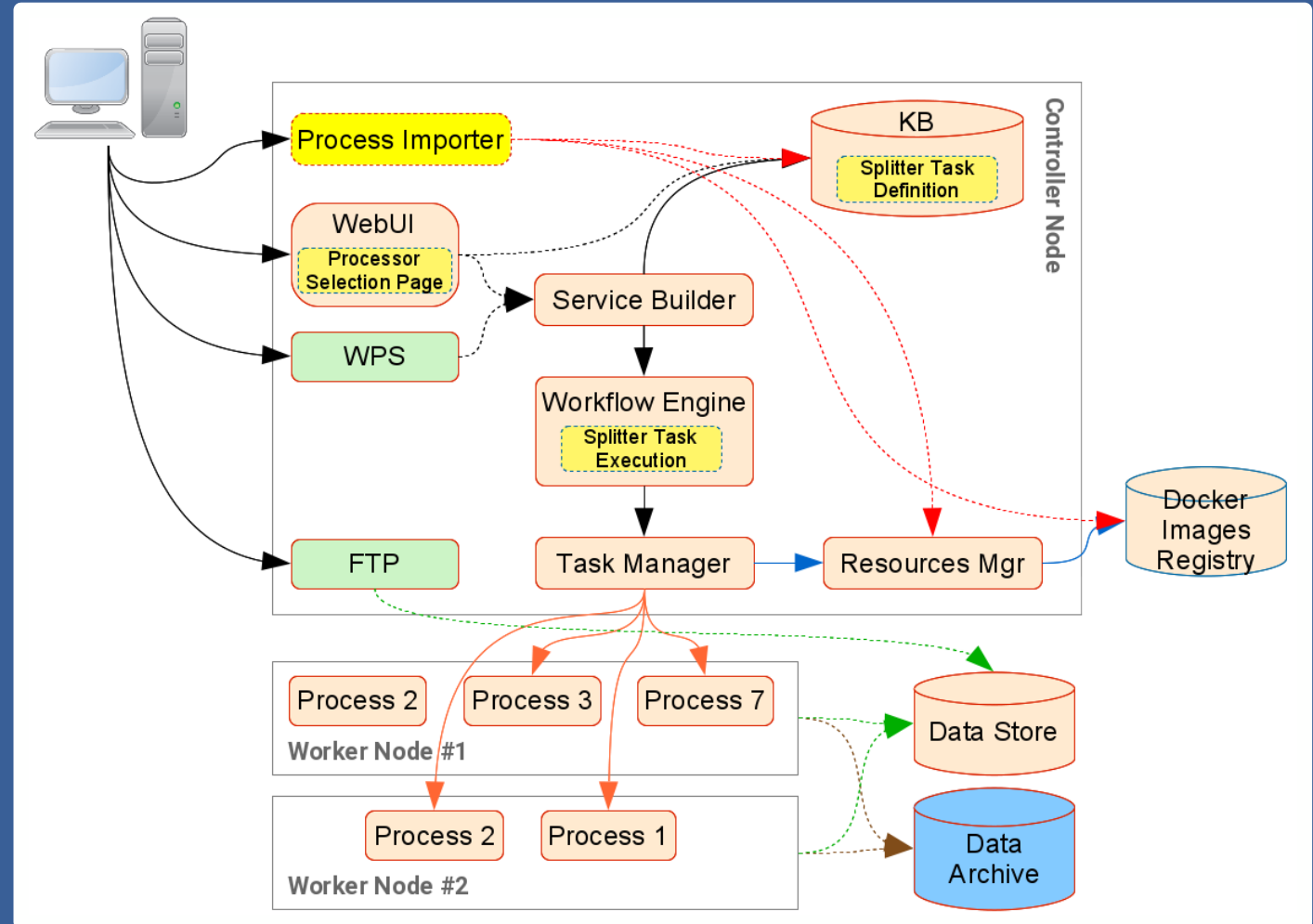
- An **Algorithm** (user provided or built-in) is packaged within a **Docker container** and referred to as a **Process** when imported in ASB.
  - ASB now includes a **Process Import Tool** for packaging and importing custom algorithms.
- A **Process** has (typed) **input and output parameters**.
  - Input parameters are above and Output parameters are below.
- A **Processor** is an application that is linked to a **Mission**, or Project, and is defined with a **Workflow** of Processes.
  - ASB includes a graphical **Workflow Editor** for creating workflows interactively, e.g. connecting parameters with drag-n-drop.
- Processors can be executed on-demand or scheduled.
- A Processor can be executed many times with the same or different parameters.
  - ASB includes interfaces for monitoring and control, reporting and data access.



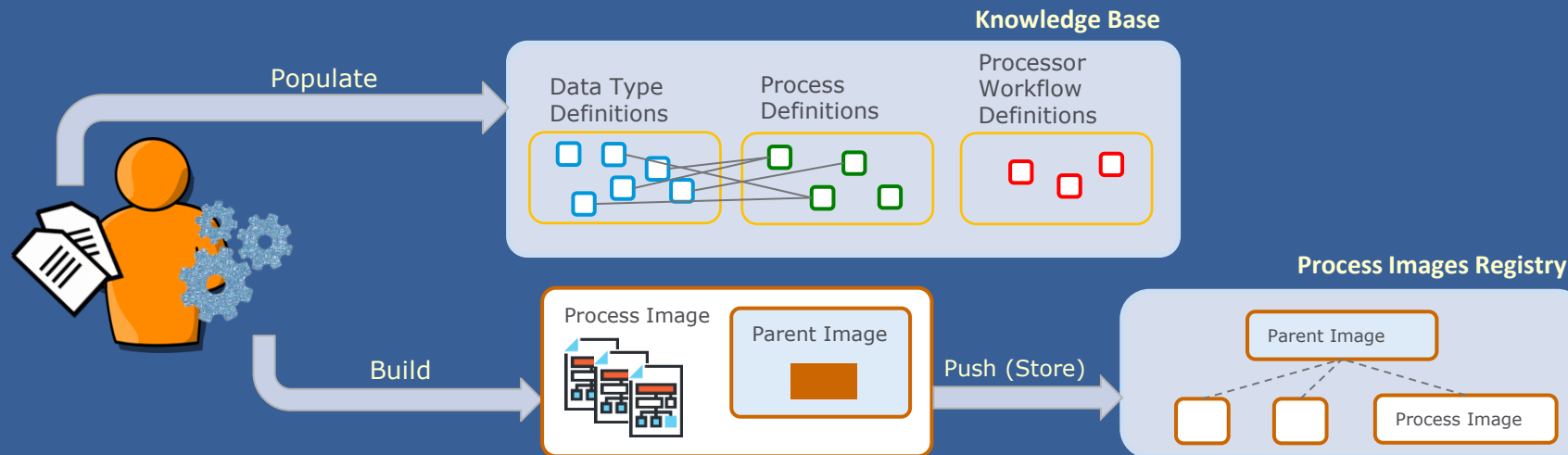
## Implemented extensions

- Process Import Tool
- Parallelisation of large collections of data using Splitter tasks
- Processor selection and execution

The impact of the new features on the overall architecture is highlighted in yellow:



- A user has a processing chain description and the algorithms that implement the different steps of the chain.
- The processing chain description is used to add knowledge into the framework: data types, parameters, processes and processor/workflow definitions.
- The algorithms are packaged within Docker container images and are pushed in a Docker registry.





## Process Importer Homepage

### Process Wrapper Template Generator

A Process Wrapper is a function written in the Python scripting language that accompanies the algorithm and acts as a connector between the process external interface (OGC WPS-based API) and your algorithm.

At execution time, the wrapper function receives the process inputs via function parameters, executes the algorithm, and returns the outputs.

The generated wrapper function is a template because it must still be your actual algorithm.

such a manner that your inputs and the function returns

[Generator](#)

### Process Algorithm Importer

Use the Process Algorithm Importer tool to upload the completed Process Wrapper, your algorithm (scripts or binaries), configuration files, and any additional dependencies.

The tool will package your algorithm and test its ingestion into the platform (dry-run import).

In case of success, you will have to possibility to either confirm the import or abort the procedure.

Imported algorithms become available as processes in the platform for integration in processor workflows.

Note: If not done yet, use the Process Wrapper Template Generator (on the left) to download and customize a Process Wrapper.

[Process Algorithm Importer](#)

### Process Wrapper Template Generator

Identification

Name

ASB CCN2 Acceptance 1

Description

Process description

Version

3

Author

Process author

Mission

Example

Input Parameters

Name	Data Type	Default value
input 1	User Integer	
input 2	User Integer	

Output Parameters

Name	Data Type
Total	User Integer
Difference	User Integer

Software Dependencies

Languages / Interpreters

☒ Python 2  
☐ Python 3  
☐ R  
☐ Oracle JRE Bu171 64bit  
☐ OpenJDK Headless 1.8.0 64bit

Tools / Libraries

☐ GDAL (including bindings for selected languages)  
☐ SNAP + Sentinel Toolbox v6.0  
☐ SNAP + SMOS Toolbox v6.0  
☐ GeoTuples 1.1.6 (requires Oracle JRE)  
☐ LST TSA Python Toolbox (requires Python 3)

Processing Resources

RAM

1GB 2GB 4GB 8GB 16GB

Disk

1GB 10GB 100GB 500GB 1TB

vCPU / Cores

1 2 4 6 8 16

```

Process Wrapper Template

#!/usr/bin/python

# -----
# Save this code in file "process_wrapper.py" and adapt as indicated in inline comments.
# -----
# Notes:
# - This is a Python 2 script. It is incompatible with Python 3.
# - The inputs will be given values by name, thus their order has no importance ...
# - ... except that the inputs with a default value must be listed last.
# - Parameter names are automatically converted into valid Python variable names.
# - Any empty line or line starting with a '#' character will be ignored.
# -----

SNAP_HOME="/home/workers/snap/"
JAVA_HOME="/usr/local/jre"
JRE_HOME="/usr/local/jre"
GEOTUPLES_HOME="/usr/local/geotuples/"

def execute(out_dir, input_1, input_2):
    """
    Identification:
    Name -- ASB CCN2 Acceptance 1
    Description --
    Version -- 3
    Author --
    Mission -- example

    Inputs:
    input_1 -- Input 1 -- 46/Byte Integer
    input_2 -- Input 2 -- 46/Byte Integer

    Outputs:
    total -- Total -- 46/Byte Integer
    difference -- Difference -- 46/Byte Integer

    Software Dependencies:
    python-2

    Processing Resources:
    ram -- 4
    disk -- 10
    cpu -- 1
    """
    total = None
    difference = None

    # -----
    # Insert your own code below.
    # Store the generated files in the "out_dir" folder. Anything else is dropped after
    # each process execution.
    # Give appropriate values to the output parameters. These will be passed to the next
    # process(es) following the workflow connections.
    # -----

    # ...

    # The wrapper must return a dictionary that contains the output parameter values.
    # -----
    return {
        'total': total,
        'difference': difference
    }
  
```

### Process Importer - Process Files Upload

#### Process Files Dropzone

Drop files here to upload

 The process properties are valid. Make sure all your process files are uploaded then click on the "Accept and Proceed" button, below.

#### Process Information

Not the properties you expected? Edit the process\_wrapper.py file and upload it again. You will immediately see updated properties below.

#### Process Files

Process files

- process\_wrapper.py (1962 bytes)
- python\_wrapper.impl (1469 bytes)
- pywps\_process.py (913 bytes)

#### Generated Dockertile

This document is automatically generated using the process properties.

[Remove Selected Files](#) [Dry Run Process Import](#)

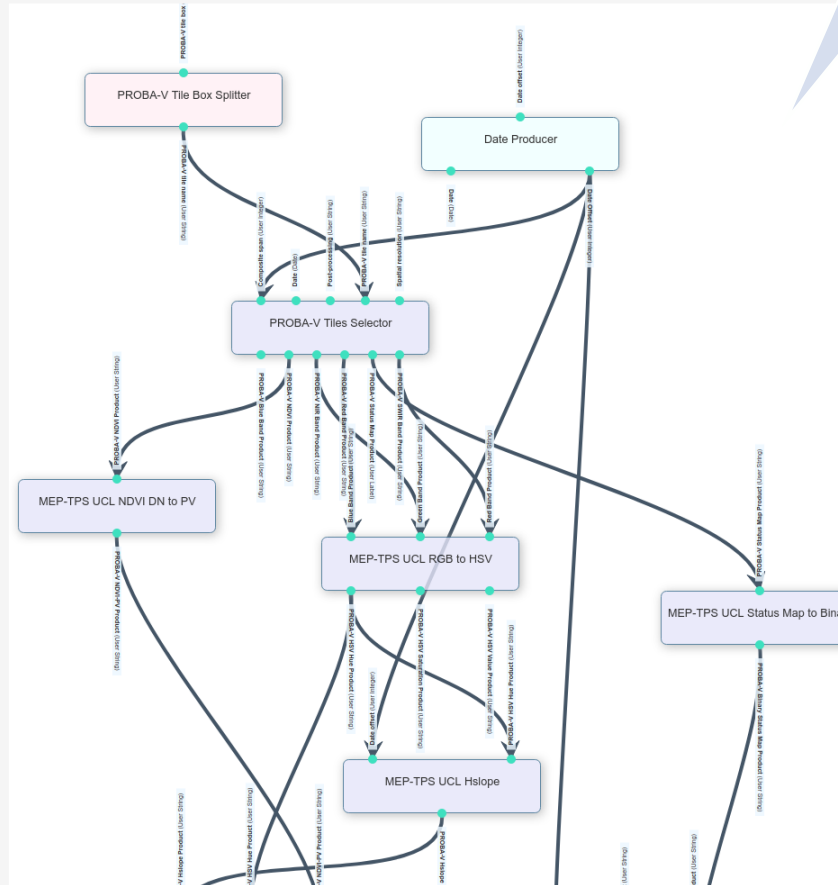
Available  
Processes

[UCL] Desert Locust Habitat Dynamic Greenness Map

Add Process

Filter ...

- 52°North Test Algorithm  
Mission: OGC EOEP Hackathon 2018
- Add Label to Image  
Mission: Generic
- BBox Converter  
Mission: Generic, PROBA-V
- Blur Image  
Mission: Generic
- Burned Areas Detector (52N)  
Mission: OGC EOEP Hackathon 2018
- Calculate Composite  
Mission: PROBA-V
- Custom Process 1  
Mission:
- Custom Process 1  
Mission:
- Date Producer  
Mission: Generic
- Date Range Converter  
Mission: Generic, PROBA-V
- Date Range Producer  
Mission: Generic
- Email Notification  
Mission: Generic
- EOEPH18 Thales Alenia Space Test Process  
Mission: OGC EOEP Hackathon 2018
- Generic FTP Get  
Mission: Generic
- Gen FTP Get



Workflow Drawing  
Canvas

Selected Process  
Instance Details

Workflow Task Details

MEP-TPS UCL Timemeter DGM

Save changes

Input Parameters

Date offset

Label: Date offset  
Default: 10

Input field: ☒ Visible ☒ Editable

PROBA-V All Vegetation Product

PROBA-V All Vegetation Corrected Product

Label: PROBA-V All Vegetation Product  
Default:

Input field: ☒ Visible ☒ Editable

PROBA-V Binary Status Map Product

Label: PROBA-V Binary Status Map Product  
Default:

Input field: ☒ Visible ☒ Editable

Output Parameters

PROBA-V DGM Product

PROBA-V Dynamic Greenness Map Product

Customizable  
Input Properties

Parameterization  
Form Preview

User Form Preview

PROBA-V tile box X17Y04/X17Y05

Date offset 10

Date

Post-processing TOC

Spatial resolution RES\_1KM

\* Not user editable

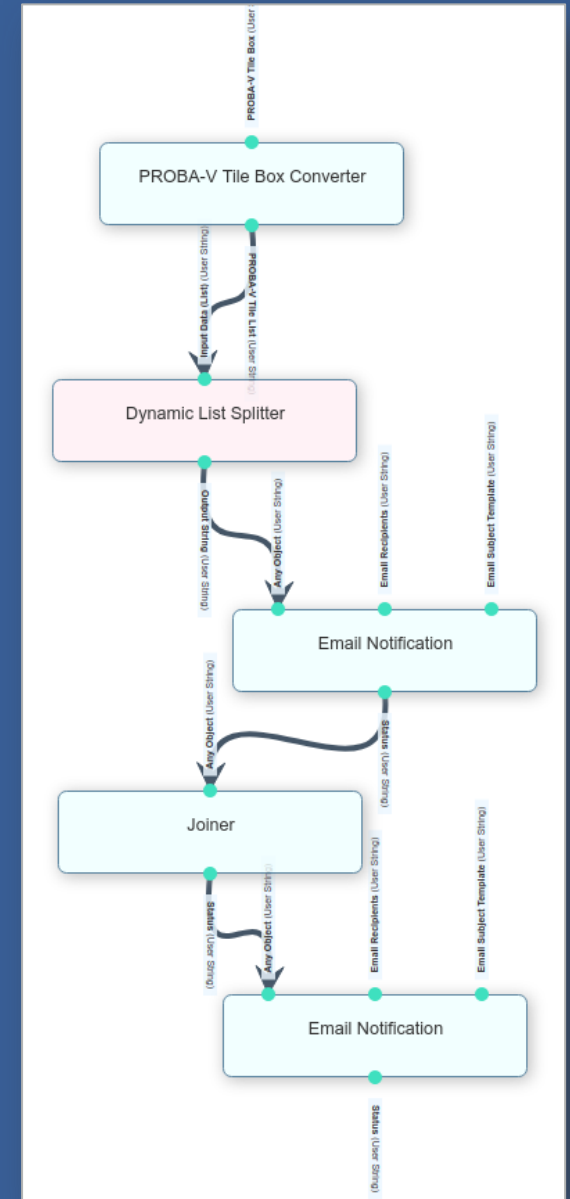
# PARALLELISING LARGE COLLECTIONS OF DATA USING SPLITTER TASKS (1/2)

## Dynamic Input List Splitter Task

- Receive a list of items as input parameter
- Execute the next tasks in the chain (workflow fragment) once for each item in the list
- Execute workflow fragments in parallel

## Enhancement

- Added support for dynamically fetched or generated lists
- Allows integrating a Dynamic List Splitter in-between the tasks in a workflow
- In this example:
  - The PROBA-V Tile Box Converter receives a tile box as input (e.g. "X16Y04/X20Y00") and outputs the corresponding list of tiles
  - The Dynamic List Splitter receives the tiles list and executes the next task (Email Notification) once on each tile
  - The Joiner task waits for all the parallel workflow fragments to complete
  - A final Email Notification is issued



# PARALLELISING LARGE COLLECTIONS OF DATA USING SPLITTER TASKS (2/2)

## Execution Reports

Product Catalogue Processors Schedules Recent Orders Orders History Credits Status Debug Information

### Product Order Execution Report

Requesting user	test
Order date and time	2018-09-05 13:46:12
Project	Example
Processor	ASB CCN2 DynSplitter Processor 3
Processor version	1
Execution credits	0
Execution start / end	2018-09-05 13:47:24 / 2018-09-05 13:49:03 Duration: 0 01:36
Status	success
Outputs	ASB Datastore (Opens in a new page)

### Input Parameters

PROBA-V Tile Box	X17Y04/X17Y06
In Progress Email Recipients	sdtest@spaceapplications.com
Email Subject Template	[ASB-CCN2] Tile [1] in progress
Final Email Recipients	sdtest@spaceapplications.com
Email Subject Template	[ASB-CCN2] All done

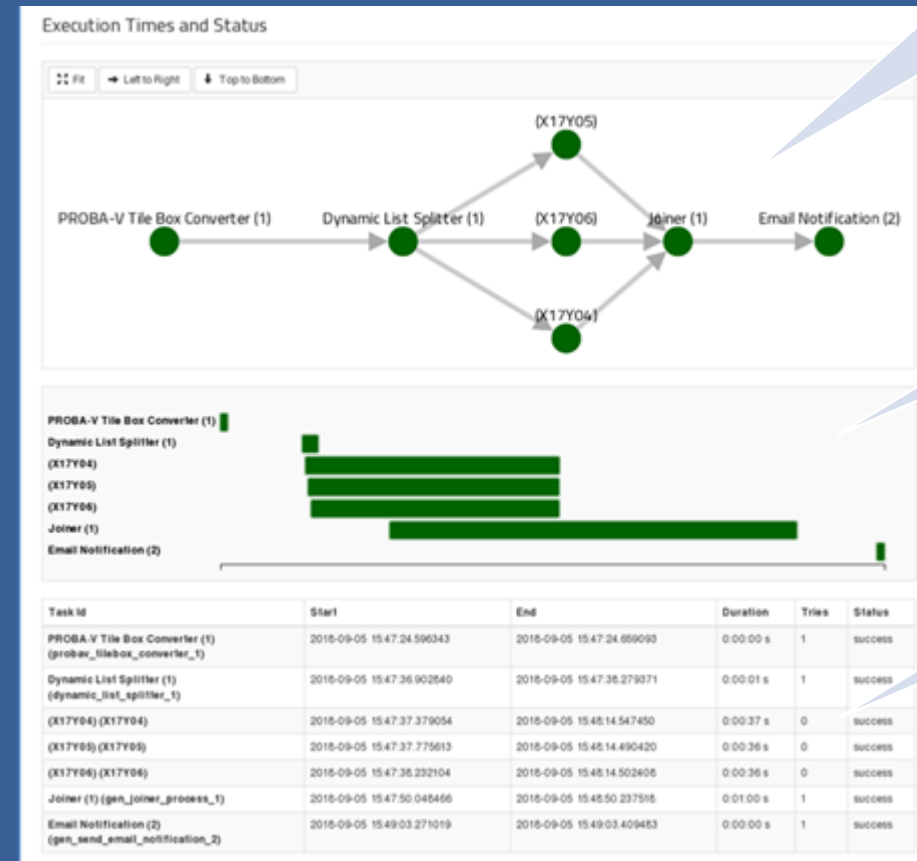
### Task Output Values

Task Id	Key	Value
Dynamic List Splitter (1)	created_dagrun_key	[35,36,37]
Joiner (1)	status_string	success
Email Notification (2)	status_string	success
PROBA-V Tile Box Converter (1)	probav_tile_box	X17Y04/X17Y06
PROBA-V Tile Box Converter (1)	probav_tile_list	["X17Y04","X17Y05","X17Y06"]

### Execution Times and Status

Top

Bottom



Graphical representation of the workflow including the fragment instances

Gantt chart showing the start/end times

Detailed start/end times of the tasks and fragments

Mission, Processor,  
Version Tree

Product Catalogue Processors Recent Product Orders Product Orders History Credits Status ☐ Debug Information

**Available Processors**

- Generic
- PROBA-V MEP TPS
- PROBA-V
- Sentinel-1
  - Sentinel-1 Change Detection
    - Version BBox Version
    - Version AOI Version**
- SMOS

[Configure and execute](#)

**Processor Description**

Project / Mission	Sentinel-1
Processor	Sentinel-1 Change Detection
Version	AOI Version
Creator / Provider	
Creation date	
Description	This version of the processor takes a WKT geometry as input.

Selected Processor  
Details

Product Catalogue Processors Recent Product Orders Product Orders History Credits Status

**Products Generation Form**

Change Detection Algorithm S1CDS Change Detection Algorithm

Region Sayam refugee camp

Sentinel-1 polarisation mode VV+VH

Sentinel-1 Scene ID S1B\_IW\_GRDH\_1SDV\_20171013T061708\_20171013T061733\_007807\_00DCA4\_7212

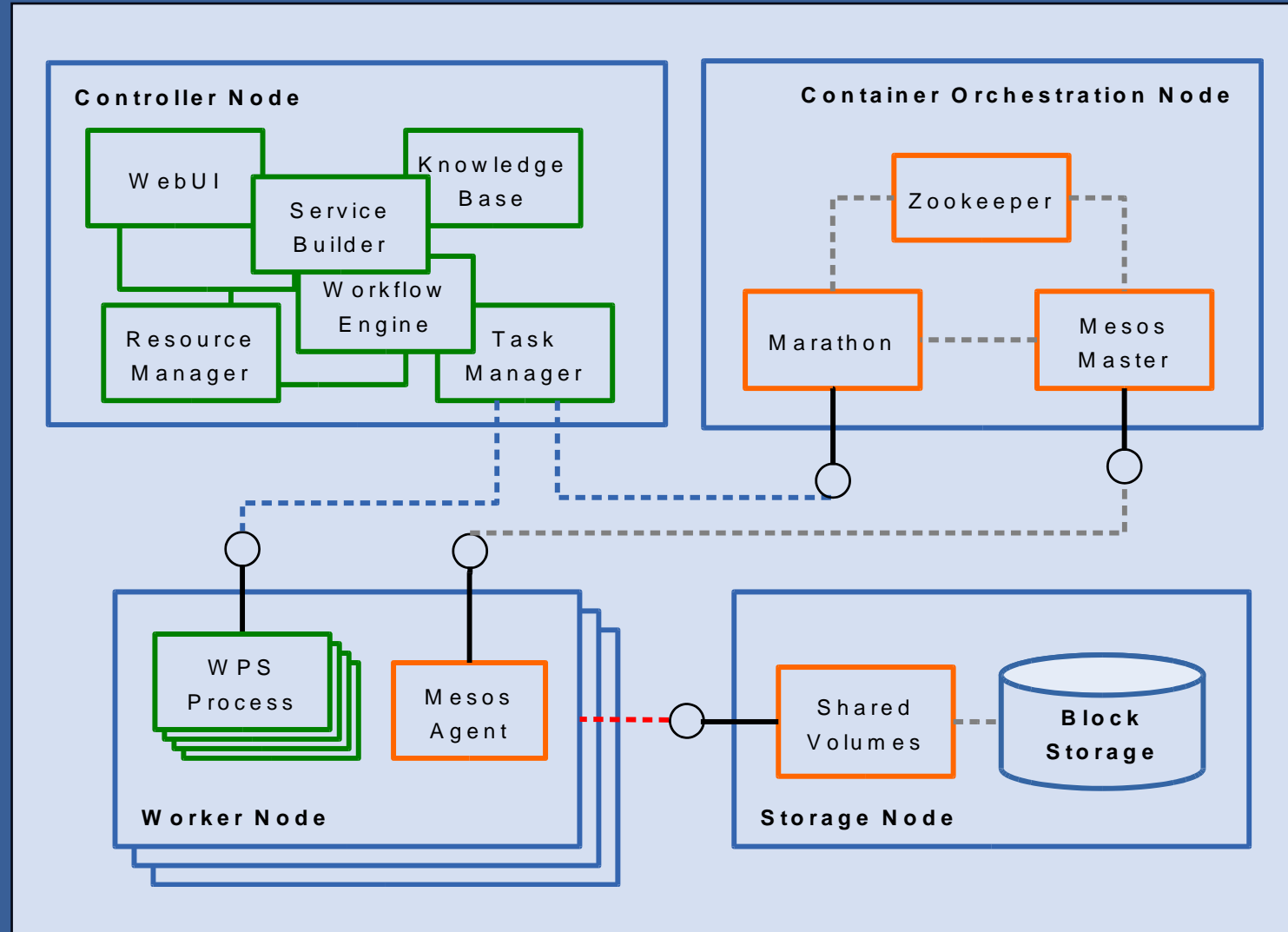
[Submit the Product Order](#)

Parameterization  
Form

Submit Button

- **Controller Node (1)**  
ASB Core Components
- **Orchestration Node (1)**  
Deployment, execution, un-deployment of Docker Containers
- **Worker Nodes (3)**  
Execution of WPS Processes within the deployed Docker Containers
- **Storage Node (1)**  
Shared access and storage of data files
- **Docker Registry (1, not represented)**

Tested and demonstrated in public cloud environments: CloudSigma, OTC Cloud, Hetzner



## 1. Workflow Editor

- Available processes, canvas, parameters customization, form preview

## 2. Process Import Tool

- Process Wrapper Template Generator
- Import a custom process wrapper with a call to an R script

## 3. Workflow Editor

- Creation of a new Processor
- Insertion of the new Process in the Processor Workflow

## 4. End-User Interface

- Selection of the new Processor, parameterization, execution
- Access to the Processor execution report





### **Collects the material produced within the CEN2 development lifecycle:**

1. Use Case Scenarios denoting the user needs
2. Technical Specifications derived from the use case scenarios
3. Updated architecture and design of the new features
4. Software verification and validation test plan, including acceptance test procedures
5. Acceptance test report
6. CEN2 features usage instructions (software user manual)
7. Demonstration plan
8. Software configuration file material (against the main ASB SCF)

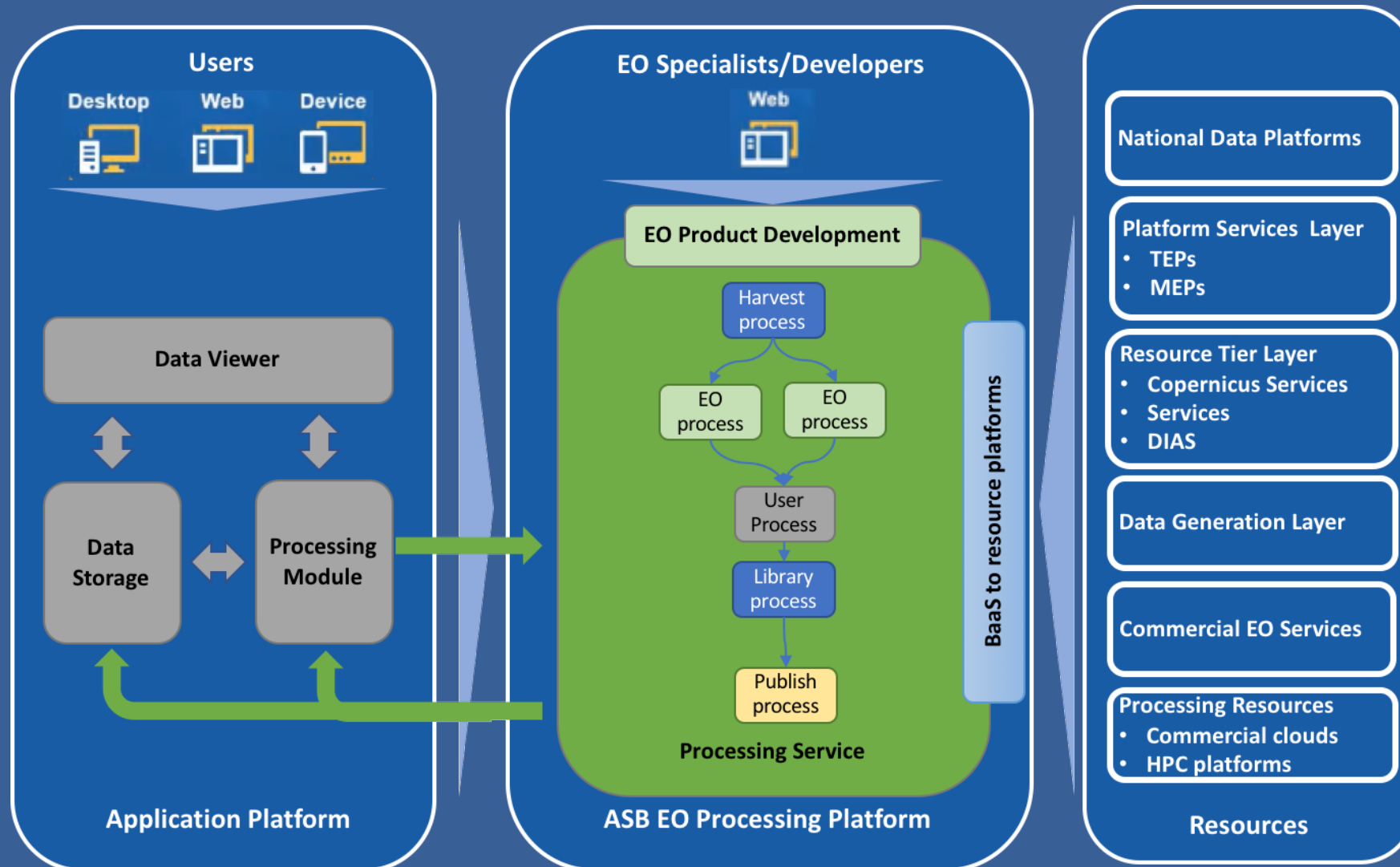
### **Releases delivered at DDR and AR**

## Demonstration Activities

- The key capabilities of the ASB Framework have been presented to technical partners in EC's H2020 funded project EOPEN. This has happened during the first EOPEN Plenary Meeting organized in Athens on September 13 and 14, 2018.
- PROBA-V MEP TPS partner LIST has been invited to attend a demonstration of the new capabilities of the ASB Framework via a teleconference. In particular, it has been shown to LIST how R functions they have implemented in the PROBA-V Time Series Analysis (TSA) toolbox can be imported within ASB and chained within workflows.

**Feedback collected in the Consolidated Evaluation Report, delivered before the Final Presentation**

- The goals of CCN2 have been achieved providing a framework for processing complex processing chains for on-demand and automated (unsupervised) processing services.
- We are seeing, after a lot of effort, a pick-up on ASB. There are misconceptions to overcome.
- The ASB Framework has achieved a major goal of being agnostic to the platform. We have demonstrated that you can easily switch from one platform to another without changing the implemented processing services.
- Low code, no IT knowledge needed to import user modules.
- Users find it easy to run processing services using dynamically generated forms.
- ASB is reaching project status and is being used in services for SatCen and the H2020 EOPEN.
- In EOPEN application users will use processing services transparently from their GIS environment.



- Tackle the user "wish list"
  - bulk imports to build a library of modules (LIST PROBA-V Toolbox, SNAP toolboxes...)\*\*
  - improved monitoring and control of process executions
  - segmenting long processing chains for optimizing the packaging and the executions
- Linking or embedding such tools as Jupyter Notebook seems to be the next step to support processing chain development making development work truly seamless
- Document concerns on performance and address them by performing a PDGS IPF realistic example
- Extend with new built-in generic processes in support of building processing chains
- The ASB Framework is ready for use in operational systems application to compare capabilities of ASB to those needed for new missions for PDGS IPFs
- Investigate over the ASB Framework potential with the needs of the Common Architecture
- Demonstrate a use of the Network of EO Resources

\*\* Already investigated direct calls to SNAP functions and use in a processing chain.

# Thank you

Bernard Valentin

[Bernard.valentin@spaceapplications.com](mailto:Bernard.valentin@spaceapplications.com)

Leslie Gale

[Leslie.gale@spaceapplications.com](mailto:Leslie.gale@spaceapplications.com)