TEP Architecture & Standards Evolution

Salvatore Pinto

30/06/2015
Outline

- TEP Architecture
- Standards requirements and evolution
Exploitation Platforms Project

ESA “earth Science Application Exploitation Platform” project, for the building and pre-operations of platform to foster exploitation of Earth Observation and earth science related data

- **6 Thematic Exploitation Platforms** sub-projects (hydrology, forestry, polar, urban, costal, geo-hazard)
- **Open Architecture** (built on collaboration between ESA and thirty parties), harmonised among the EPs
- Open interface **standards and profiles** (HMA, OGC)
- Resources **interoperability**, between EPs and other projects
- Common Core Components (**Open Source**, with the aim to maximise reuse of existing Open Source project)
What is an Exploitation Platform?

For eSAEP, an Exploitation Platform is a virtual workspace, with the final aim to **bring the users to the data**.

The Exploitation platform is providing a user community with

- **large volume of data** (EO and non-space data)
- an algorithm development and **integration environment**
- **processing services** and software (e.g. toolboxes, retrieval baselines, visualization routines)
- **computing resources** (e.g. hybrid cloud/grid)
- **collaboration tools** (eg. forums, wiki, knowledge base, open publications, social networking...)

- general operation capabilities (e.g. user management and access control, accounting, billing, etc...).
The **Exploitation Platform Open Architecture** definition is an on-going project, building on experience from the TEP and related projects. Currently, the **DRAFT2** has been released with Creative Commons Attribution-Share Alike 4.0 International License.
Key points in EP Architecture Design

1. Clear separation between functionalities (processing, data search, data access)
2. Interoperability and standards
3. ICT infrastructure independence
4. Cost-effective ICT provisioning
5. Open Source software, Open Standards, Collaboration
6. User friendly interfaces (for all functionalities, including software integration, etc…)
7. Common authentication framework (EO-SSO)
8. Pay-per-use (accounting, quota management, sponsorship)
## Key components of an EP

According to the current EP Architecture draft, the key common core user-facing components of an Exploitation Platform are listed in the table (with description and related functionality):

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalogue</td>
<td><strong>Resource search</strong> interface, for user to look for platform resources (eg. data, results, processing services), according to metadata, classification, ranking, etc...</td>
</tr>
<tr>
<td>Resource Access Gateway</td>
<td>Manage the <strong>Resources access</strong>, for the platform. Controls users access to binary resources representation (e.g. input EO data products) and provides access interface.</td>
</tr>
<tr>
<td>Execution Gateway</td>
<td>Provides the <strong>Processing service execution</strong> interface for submission of processing requests.</td>
</tr>
<tr>
<td>Workflow/App Manager</td>
<td>Imports, interprets and executes a service represented by a <strong>Processing service package</strong>.</td>
</tr>
<tr>
<td>Geo Resource Browser</td>
<td>Provides the possibility to display all resources involved in a processing (eg. data, results, processing services), package them into <strong>Processing Containers</strong> and replicate processing</td>
</tr>
<tr>
<td>AAI</td>
<td>Manage the <strong>Authentication ad Authorization Infrastructure</strong> for the users of the platform.</td>
</tr>
</tbody>
</table>
Full components list

**Workspace**: Provide functionalities to the user to manage processing services

**Collaboration Bucket**: provides a framework for sharing unvalidated processing services, processing results, etc...

**Accounting and monitoring**: provides a central integrated accounting and monitoring system

**Execution Cluster**: Provides the environment and ICT resources where processing services are executed (for normal operations or testing)
Outline

- TEP Architecture
- Standards requirements and evolution
Key points for the standards selection in the EP:

- **Web/developer oriented API**
- M2M communication
- Simple client
- Efficient encoding

Which translates to preference to the following technologies:

- **JSON over XML**
- **REST over SOAP**
- Pre-defined namespace vs custom namespaces

Moreover, standards shall contain Best Practices for Implementation, with:

- Clear definition of **namespaces, nomenclature and conventions**
- Definition of End-to-end Scenario involving the interface/product
- **Samples for all messages exchange** in the end-to-end scenario
For the EP, in conjunction with HMA/OGC, we plan to evolve the existing standards to address the new EP, Big Data and Web 2.0 requirements.

The current baseline for this evolutions is:

<table>
<thead>
<tr>
<th>Implementation BP</th>
<th>Based on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource search</td>
<td>OGC OpenSearch standards and BPs</td>
</tr>
<tr>
<td>Resource access</td>
<td>OGC Download Services for EO</td>
</tr>
<tr>
<td>Processing service execution</td>
<td>OGC WPS</td>
</tr>
<tr>
<td>Processing service packaging</td>
<td>BPEL/XPDL</td>
</tr>
<tr>
<td>Processing container</td>
<td>OGC OWS Context</td>
</tr>
<tr>
<td>AAI</td>
<td>OGC User Management Interfaces for EO</td>
</tr>
</tbody>
</table>
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, eg. processing services, insitu data, processing results
4. New attributes for data ranking and query
5. Additional cost attribute
6. Extended Resource Location System (manage multiple file locations, selecting the best according to the request)
7. Add management interface (data ingestion, data edit, data delete)
8. Standardize data preview format for geo-coded data (e.g. via KML)
9. Add catalogue statistics (spatial, temporal coverage and custom statistics)
10. Port “CEOS OpenSearch Best Practices Document” to the OGC
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, eg. processing services, insitu data, processing results

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Occurrences</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>application:id</td>
<td>Unique identifier of the application. Mandatory.</td>
<td>minOccurs=1</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxOccurs=1</td>
<td></td>
</tr>
<tr>
<td>application:name</td>
<td>Name of the application. Mandatory.</td>
<td>minOccurs=1</td>
<td>String</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxOccurs=1</td>
<td></td>
</tr>
<tr>
<td>application:description</td>
<td>A short description of the application. Optional.</td>
<td>maxOccurs=1</td>
<td>String</td>
</tr>
<tr>
<td>application:abstract</td>
<td>A description of the application in terms input output, minimum requirements to run the application. Optional.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>application:owner</td>
<td>Link to the OSSD of the person, institute, company (see contactInfo) who own the application. Mandatory</td>
<td>minOccurs=1</td>
<td>anyUri</td>
</tr>
<tr>
<td>application:publisher</td>
<td>Link to the OSSD of the person, institute, company (see contactInfo) who has published the application on the platform. Mandatory</td>
<td>minOccurs=1</td>
<td>anyUri</td>
</tr>
<tr>
<td>application:addedon</td>
<td>Date when the application has been uploaded to the platform. Mandatory.</td>
<td>minOccurs=1</td>
<td>dateTime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxOccurs=1</td>
<td></td>
</tr>
<tr>
<td>application:lastupdate</td>
<td>Date of the last update of the application. Mandatory.</td>
<td>minOccurs=1</td>
<td>dateTime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxOccurs=1</td>
<td></td>
</tr>
<tr>
<td>application:publications</td>
<td>Link to the OSSD of the publications related to the application</td>
<td>minOccurs=1</td>
<td>anyUri</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxOccurs=1</td>
<td></td>
</tr>
<tr>
<td>application:dependency</td>
<td>Link to the OSSD of the list of all the applications which are required to run the current application (see application)</td>
<td>minOccurs=0</td>
<td>anyUri</td>
</tr>
</tbody>
</table>
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, eg. processing services, in-situ data, processing results
4. New attributes for data ranking and query

```xml
<dc:title>SIR_SIN_FR</dc:title>
<atom:link rel="search" type="application/opensearchdescription+xml" title="Search the SIR_SIN_FR"
  href="http://grid-eo-catalog.esrin.esa.int/catalogue/gpod/SIR_SIN_FR/description/"/>
<dc:creator>CRYOSAT</dc:creator>
<dc:format>CRYOSAT</dc:format>
<rank:tag>Volcano_Earthquake_Monitoring</rank:tag>
<rank:numberOfViews>352</rank:numberOfViews>
<rank:numberOfShares>3000</rank:numberOfShares>
<rank:votes>4 stars</rank:votes>
<rank:fieldOfInterest>Geohazards_Hydrology</rank:fieldOfInterest>
<dc:extent>17736</dc:extent>
</dc:title>
```
OpenSearch
Evolution of Resource Search standard

Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, eg. processing services, insitu data, processing results
4. New attributes for data ranking and query

5. **Additional cost attribute**

```
<dclite4g:DataSet rdf:about="http://grid-eo-catalog.esrin.esa.int/catalogue/gpod/SIR_SIN_FR/CS_OFFL_SIR_SIN_FR_20150524T080232_20150524T080343_C001.rdf">
  <dc:identifier>CS_OFFL_SIR_SIN_FR_20150524T080232_20150524T080343_C001</dc:identifier>
  <dclite4g:series rdf:resource="http://grid-eo-catalog.esrin.esa.int/catalogue/gpod/SIR_SIN_FR.rdf"/>
  <dclite4g:onlineResource>
    <ws:GRIDSITE rdf:about="https://store22.esrin.esa.int/DAT/SIN_FR_20150524T080232_C001.TGZ" ws:preference="50"/>
  </dclite4g:onlineResource>
  <eop:orbitNumber>27163</eop:orbitNumber>
  <eop:size>196885479</eop:size>
  <eop:trackNumber>2056</eop:trackNumber>
  <ical:dtstart>2015-05-24T08:02:32.000Z</ical:dtstart>
  <ical:dtend>2015-05-24T08:03:43.000Z</ical:dtend>
  <eop:processingDate>2015-06-24T18:25:55.000Z</eop:processingDate>
  <dct:cost>10</dct:cost>
  <dct:costUnit>euro</dct:costUnit>
  <dclite4g:modified>2015-06-29T13:04:44.183Z</dclite4g:modified>
  <dclite4g:created>2015-06-29T13:04:44.183Z</dclite4g:created>
</dclite4g:DataSet>
```
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, eg. processing services, insitu data, processing results
4. New attributes for data ranking and query
5. Additional cost attribute

6. **Extended Resource Location System** (manage multiple file locations, selecting the best according to the request)

```xml
<dc:identifier>CS_OFFL_SIR_SIN_FR_20150524T080232_20150524T080343_C001/01</dc:identifier>
<dc:identifier>CS_OFFL_SIR_SIN_FR_20150524T080232_20150524T080343_C001</dc:identifier>
<dc:identifier>http://grid-eo-catalog.esrin.esa.int/catalogue/gpod/SIR_SIN_FR/01/</dc:identifier>
<dc:identifier>http://grid-eo-catalog.esrin.esa.int/catalogue/gpod/SIR_SIN_FR/01/</dc:identifier>
<dc:identifier>http://store22.esa.int/DAT/SIN_FR_20150524T080232_C001.TGZ</dc:identifier>
<dc:identifier>http://store22.esa.int/DAT/SIN_FR_20150524T080232_C001.TGZ</dc:identifier>
<dc:identifier>https://store22.esa.int/DAT/SIN_FR_20150524T080232_C001.TGZ</dc:identifier>
<dc:identifier>https://store22.esa.int/DAT/SIN_FR_20150524T080232_C001.TGZ</dc:identifier>

**NOTE:** ws:preference shall be generated according to the source IP and other metrics
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, e.g. processing services, insitu data, processing results
4. New attributes for data ranking and query
5. Additional cost attribute
6. Extended Resource Location System (manage multiple file locations, selecting the best according to the request)

**NOTE: HTTP RESTful, e.g.:** PUT /{Product URI}/atom

7. Add management interface (data ingestion, data edit, data delete)
OpenSearch
Evolution of Resource Search standard

Proposed evolution for the OpenSearch standard:
1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, e.g. processing services, insitu data, processing results
4. New attributes for data ranking and query
5. Additional cost attribute
6. Extended Resource Location System (managing multiple file locations, selecting the best according to the request)
7. Add management interface (data ingestion, data edit, data delete)
8. **Standardize data preview format for geo-coded data (e.g. via KML)**
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, e.g., processing services, in situ data, processing results
4. New attributes for data ranking and query
5. Additional cost attribute
6. Extended Resource Location System (manage multiple file locations, selecting the best according to the request)
7. Add management interface (data ingestion, data edit, data delete)
8. Standardize data preview format for geo-coded data (e.g., via KML)
9. Add catalogue statistics

Evolution of Resource Search Standard

- Get Query Coverage Stats: Coverage statistics
- Get Temporal Coverage Stats: Coverage statistics
- Get Spatial Coverage Stats: Coverage statistics

Client (Management)

Catalogue

Selective harvesting
Proposed evolution for the OpenSearch standard:

1. Add JSON rendering with support to JSON-LD
2. Detail sync function for metadata harvesting
3. Define non-EO data resource classes, e.g. processing services, insitu data, processing results
4. New attributes for data ranking and query
5. Additional cost attribute
6. Extended Resource Location System (manage multiple file locations, selecting the best according to the request)
7. Add management interface (data ingestion, data edit, data delete)
8. Standardize data preview format for geo-coded data (e.g. via KML)
9. Add catalogue statistics (spatial, temporal coverage and custom statistics)
10. Port “CEOS OpenSearch Best Practices Document” to the OGC
OGC Download Service for EO Products BP
Evolution of Data Access standard

Proposed evolution for the Data Access best practice

1. Clarify univocally when the product is a metalink (e.g. from Product URI)
2. Add JSON and HTML rendering of Accepted Download Response
3. Quota management (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)
4. Extended Authorization error handling (authorization failure reason, user sign TOC URL, authorization request URL, etc...)
5. Download statistics
Proposed evolution for the Data Access best practice

1. Clarify univocally when the product is a metalink (e.g. from Product URI)

   e.g. in this case, product-URI shall end with /:
   GET /{Product URI}/
Proposed evolution for the Data Access best practice

1. Clarify univocally when the product is a metalink (e.g. from Product URI)

2. Add JSON and HTML rendering of Accepted Download Response

```xml
<xml version="1.0" encoding="UTF-8"?>
<dse:ProductDownloadResponse
xmlns:dse="http://www.opengis.net/spec/DSEO/1.0/schema/ProductDownloadResponse
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/spec/DSEO/1.0/schema/ProductDownloadResponse"
   >
   <dse:ResponseCode>IN_PROGRESS</dse:ResponseCode>
   <dse:RetryAfter>30</dse:RetryAfter>
</dse:ProductDownloadResponse>
```

```json
{ ResponseCode: "IN_PROGRESS", RetryAfter: 30 }
```

```html
<html>
<head>
   <title>Repository access in progress</title>
   <meta http-equiv="refresh" content="5">
</head>
<body>
   <h3>Download in progress</h3>
   <p>The system is accessing the remote repository... Please wait...<br />This page will automatically refresh after 5 seconds...<br /></p>
</body></html>
```
Proposed evolution for the Data Access best practice

1. Clarify univocally when the product is a metalink (e.g. from Product URI)
2. Add JSON and HTML rendering of Accepted Download Response
3. Quota management (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)

**NOTE: HTTP RESTful, e.g.:** GET /{Product URI}?quota
Proposed evolution for the Data Access best practice

1. Clarify univocally when the product is a metalink (e.g. from Product URI)
2. Add JSON and HTML rendering of Accepted Download Response
3. Quota management (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)
4. **Extended Authorization error handling** (authorization failure reason, user sign TOC URL, authorization request URL, etc...)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<deo:ProductDownloadResponse
 xmlns:deo="http://www.opengis.net/spec/DSEO/1.0/schema/ProductDownloadResponse
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/spec/DSEO/1.0/schema/ProductDownloadResponse">
  <deo:ResponseCode>AUTHORIZATION_ERROR</deo:ResponseCode>
  <deo:reason>TOC_NOT_SIGNED</deo:reason>
  <deo:UserRequiredAction>http://earth.eo.esa.int/sign_TOC.php</deo:UserRequiredAction>
</deo:ProductDownloadResponse>
```
Proposed evolution for the Data Access best practice

1. Clarify univocally when the product is a metalink (e.g. from Product URI)
2. Add JSON and HTML rendering of Accepted Download Response
3. Quota management (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)
4. Extended Authorization error handling (authorization failure reason, user sign TOC URL, authorization request URL, etc…)
5. Download statistics

NOTE: HTTP RESTful, e.g.: GET /stats/{Product URI}/download
OGC Web Processing Service
Evolution of Processing service execution

Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
4. Add additional Job Control operations (pause, resume, restart)
5. Extend outputTransmission by ‘reference’ to support custom external output locations
6. Quota support (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)
7. Cost support, a-priori (eg. via SimulateEx call) and a-posteriori
8. Add version parameter to the WPS Execute Request
9. Add default values for BoundingBoxData
Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)

```
"wps:ProcessDescriptions": {  
  "-service": "WPS",
  "-version": "1.0.0",
  "-xml:lang": "en-GB",
  "-xs1:schemaLocation": "http://www.opengis.net/wps/1.0.0",
  "-xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
  "-xmlns:xlink": "http://www.w3.org/1999/xlink",
  "-xmlns:wps": "http://www.opengis.net/wps/1.0.0",
  "-xmlns:ows": "http://www.opengis.net/ows/1.1",
  "ProcessDescription": {  
    "-wps:processVersion": "1.0.0",
    "-storeSupported": "true",
    "-statusSupported": "true",
    "ows:Identifier": "MIRAVI",
    "ows:Title": "MIRAVI Geo",
    "ows:Abstract": "Service access point for the MIRAVI service. The request must spec",
    "DataInputs": {  
      "Input": [  
        {  
          "-minOccurs": "0",
          "-maxOccurs": "1",
          "ows:Identifier": "ce",
          "ows:Title": "Computing Element",
          "LiteralData": {  
            "ows:DataType": {  
              "-ows:reference": "http://www.w3.org/TR/xmlschema-2/#string",
              "#text": "string"
            },
            "ows:AllowedValues": {  
              "ows:Value": [  
                "string",
                "integer",
                "date",
                "time",
                "duration",
                "decimal",
                "float",
                "anyURI"
              ]
            }
          }
        }
      ]
    }
  }
```

Salvatore Pinto | 30/06/2015 | Slide 30
OGC Web Processing Service
Evolution of Processing service execution

Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc…)

2. **Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)**

   ```xml
   <Input minOccurs="0" maxOccurs="1">
     <ows:Identifier>wi_frame</ows:Identifier>
     <ows:Title>wi_frame</ows:Title>
     <LiteralData>
       <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">string</ows:DataType>
       <ows:AnyValue/>
       <DefaultValue>javascript:30*proj.value</DefaultValue>
     </LiteralData>
   </Input>

   <Input minOccurs="0" maxOccurs="1">
     <ows:Identifier>proj</ows:Identifier>
     <ows:Title>Projection</ows:Title>
     <ows:Abstract>Plate-carre:Plate-carre;UTM:UTM;Stereographic:Stereographic</ows:Abstract>
     <LiteralData>
       <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">string</ows:DataType>
       <ows:AllowedValues>
         <ows:Value>Plate-carre</ows:Value>
         <ows:Value>UTM</ows:Value>
         <ows:Value>Stereographic</ows:Value>
       </ows:AllowedValues>
       <DefaultValue>javascript:wi_frame.enabled=not proj.enabled;</DefaultValue>
     </LiteralData>
   </Input>
   ```
Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
4. **Add additional Job Control operations (pause, resume, restart)**
OGC Web Processing Service
Evolution of Processing service execution

Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
4. Add additional Job Control operations (pause, resume, restart)
5. **Extend outputTransmission by ‘reference’ to support custom external output locations**

---

Table 29 – Parts of the ProcessOfferingProperties Attributes structure

<table>
<thead>
<tr>
<th>Names</th>
<th>Definition</th>
<th>Data type and values</th>
<th>Multiplicity and use</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobControlOptions</td>
<td>Job control options supported for this process</td>
<td>List of supported options for process control (see Table 30)</td>
<td>One or more (mandatory)</td>
</tr>
<tr>
<td>outputTransmission</td>
<td>Supported transmission modes for output data (by value / by reference)</td>
<td>List of supported data transmission options (see Table 31).</td>
<td>One or more (mandatory)</td>
</tr>
<tr>
<td>processVersion</td>
<td>Release version of process (not processVersionType)</td>
<td></td>
<td>Zero or one (optional)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Reference</th>
<th>Reference for publishing the processing output</th>
<th>List of supported options (e.g. Google Drive, FTP server, etc...)</th>
<th>One or more (conditional)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>defined by the process description specification.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defaults to “native”.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(conditional) Include when using a different process</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>model than the native process model.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
OGC Web Processing Service

Evolution of Processing service execution

Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
4. Add additional Job Control operations (pause, resume, restart)
5. Extend outputTransmission by ‘reference’ to support custom external output locations
6. **Quota support** (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)

**Example:**

```
GET /wps/?service=WPS&Request=GetQuota&identifier=MIRAVI
```
OGC Web Processing Service
Evolution of Processing service execution

Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc...)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
4. Add additional Job Control operations (pause, resume, restart)
5. Extend outputTransmission by ‘reference’ to support custom external output locations
6. Quota support (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)
7. **Cost support, a-priori** (eg. via SimulateEx call) and a-posteriori

---

**Simulate Execute**: Simulated Cost/Quota

**Execute**: Processing Cost
OGC Web Processing Service
Evolution of Processing service execution

Proposed evolution for the Processing service standard

1. JSON rendering of processing status response (including processing logs, cost, etc…)
2. Dynamic UI parameters (e.g. scripting for default and/or constrained parameters)
3. Interactive processing support
4. Add additional Job Control operations (pause, resume, restart)
5. Extend outputTransmission by ‘reference’ to support custom external output locations
6. Quota support (daily, monthly, yearly, total, concurrent and absolute quota, quota status retrieval, set quota for users)
7. Cost support, a-priori (e.g. via SimulateEx call) and a-posteriori
8. **Add version parameter to the WPS Execute Request**
9. Add default values for BoundingBoxData

**Table 42 – Additional properties in the Execute request**

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Data type and values</th>
<th>Multiplicity and use</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>Desired response format, i.e. a response document or raw data.</td>
<td>String {raw</td>
<td>document}</td>
</tr>
<tr>
<td>mode</td>
<td>Desired execution mode.</td>
<td>String {sync</td>
<td>async}</td>
</tr>
</tbody>
</table>

+ Version of the processor
Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)
2. “WPS describe process”-like for basic job input/output/parameters definition
3. Processing graph definition (e.g. based on XPDL or BPEL, with support to different processing paradigms, such as map/reduce, array processing, etc...)
4. Support possibility to chain workflow and import/link other workflow packages
5. Includes software packaging standard definition (e.g. RPM, Docker) and wrappers
6. Includes general workflow details, such as description, versioning and authorization information.
Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. **Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)**
Processing service packaging
New standard/best practice definition

Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)

2. “WPS describe process”-like for basic job input/output/parameters definition

```xml
<LiteralData>
  <ows:DataType ows:reference="http://www.w3.org/TR/owschema-2/#string">string</ows:DataType>
  <ows:AllowedValues>
    <ows:Value>NOCOMP</ows:Value>
    <ows:Value>COMPRESS</ows:Value>
    <ows:Value>TGZ</ows:Value>
  </ows:AllowedValues>
  <DefaultValue>truc</DefaultValue>
</LiteralData>
```

Salvatore Pinto | 30/06/2015 | Slide  40
ESA UNCLASSIFIED - For Official Use
Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)

2. “WPS describe process”-like for basic job input/output/parameters definition

3. Processing graph definition (e.g. based on XPDL or BPEL, with support to different processing paradigms, such as map/reduce, array processing, etc...)

```xml
            xmlns:wpdlXPDLVersion="1.0">
  <wpdl:PackageHeader>
    <wpdl:Vendor>Together wpdl:Vendor</wpdl:Vendor>
    <wpdl:Created>2009-02-25 07:50:07</wpdl:Created>
  </wpdl:PackageHeader>
  <wpdl:DefinableHeader PublicationStatus="UNDER_TEST"/>
  <wpdl:ConformanceClass GraphConformance="NON_BLOCKED"/>
  <wpdl:Script Type="text/javascript"/>
  <wpdl:Applications>
    <wpdl:Application Id="initializeScenario">
      <wpdl:FormalParameters>
        <wpdl:FormalParameter Id="eParams" Mode="IN">
          <wpdl:DataType>
            <wpdl:BasicType Type="STRING"/>
            </wpdl:BasicType>
        </wpdl:FormalParameter>
      </wpdl:FormalParameters>
      <wpdl:ExtendedAttributes>
        <wpdl:ExtendedAttribute Name="PoolAgentClass" Value="org.gpwhdrg.sshark.
        </wpdl:ExtendedAttribute>
        <wpdl:ExtendedAttribute Name="AppName" Value="some.wfinclude.html.Behavior">
        </wpdl:ExtendedAttribute>
      </wpdl:ExtendedAttributes>
    </wpdl:Application>
    <wpdl:Application Id="initializePatient" Name="initializePatient">
      <wpdl:FormalParameters Type="NotificationBrowser Mode="TEXT">
```
Processing service packaging
New standard/best practice definition

Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)
2. “WPS describe process”-like for basic job input/output/parameters definition
3. Processing graph definition (e.g. based on XPDL or BPEL, with support to different processing paradigms, such as map/reduce, array processing, etc…)
4. **Support possibility to chain workflow and import/link other workflow packages**
Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)
2. “WPS Describe Process”-like for basic job input/output/parameters definition
3. Processing graph definition (e.g. based on XPDL or BPEL, with support to different processing paradigms, such as map/reduce, array processing, etc...)
4. Support possibility to chain workflow and import/link other workflow packages
5. Includes software packaging standard definition (e.g. RPM, Docker) and wrappers
Create a new standard or Best Practices for packaging processing services

Proposed new standard key elements:

1. Support both not-interactive processing (e.g. Workflow) and interactive processing services (e.g. App)
2. “WPS describe process”-like for basic job input/output/parameters definition
3. Processing graph definition (e.g. based on XPDL or BPEL, with support to different processing paradigms, such as map/reduce, array processing, etc...)
4. Support possibility to chain workflow and import/link other workflow packages
5. Includes software packaging standard definition (e.g. RPM, Docker) and wrappers
6. Includes general workflow details, such as description, versioning and authorization information.
Create a new Best Practices for representing Virtual Experiments as the collection of resources related to a scientific processing performed with an Exploitation Platform.

This shall

1. Represent and link all the resources related to a scientific processing:
   a. Processing services (including processing parameters)
   b. Input and output data sets
   c. Commentary annotations
   d. Validation information (to check for expected results)
   e. Qualification information (thirty party verified quality level)

2. Support DOI (via DataCite/DOI.org standard)

3. Includes the following sub-VE definitions:
   a. Product Set: only input/output data and commentary annotations
   b. Service Test: only input, processing service and validation info
OGC User Management Interfaces for EO Services
Evolution of AAI standard

Web Services (catalogue, WMS browse & maps, WPS processing, Product download,...)
Login Services (provides a token)

ApplicaFon authenticates the user by calling a login service:
ApplicaFon gets back a token.
ApplicaFon calls a service (HTTP) with the token.
...then calls another service which itself calls another service in the same IDP domain.
The token is propagated.

...which calls yet another service in another IDP domain (federation?)

Which might itself rely on other services...
(but will come back later to get the results)

Olivier Barois
26/03/2013
Slide 46

European Space Agency
1. Spare Slides
Exploitation Platforms shares resources via standard interfaces.
Exploitation Platform Concept

Collaboration

Citation

eg. [DataCite]

Results/reproducibility

Virtual Experiment packages

Results/processors sharing

Question & Answers

Salvatore Pinto | 30/06/2015 | Slide 49

ESA UNCLASSIFIED - For Official Use
Exploitation Platform Concept
Service Integration

Development Environment
(VM or standalone application)

Workflow
App

Package Tools

Test Tools

(test) Execution Environment
Satellite Data

- ESA
  Historical Missions (ERS/ENVISAT)
- DLR
  (Terrasar-X)
- SPOT Image
  (Spot)
  
Auxiliary Data

- Copernicus
  (Sentinel-1)
- ECWMF
  (environmental variables)

User Data

- Scientists
  (in-situ data, processing results)
ICT resources are decoupled from the platform, via virtualization technologies, cloud technologies and generic API support.

This means:

- Possibility to use resources on external ICT infrastructure
- Possibility to deploy the Platform on different ICT resources.
Exploitation Platform Concept
Computing resources

Multi ICT technologies support:
public/private/hybrid Cloud, Grid, Cluster

• Flexible deployment
• Multi API standard, single client library (eg. libcloud, Globus)
• Command-line and GUI applications execution (OS independent)
Cost-effectiveness (resource usage optimization)

Cost Effectiveness:

\[
\frac{\text{Resources spent for Scientific Processing}}{\text{Total amount of resources spent}} \rightarrow 1
\]

- Use small-size packages
- Use shared resources
- Pre-install processors packages
- Cache the data

Total amount of resources spent
OSS Best Practices

Exploitation Platforms components follow **Open-Source Software development best practices** for implementation (e.g. code on a public repository, small and frequent commits, manage branches, do not fork dependencies, evaluate merge pull requests)

GitHub

[jjrom / resto2](https://github.com/jjrom/resto2)


- 489 commits
- 2 branches
- 2 releases
- 3 contributors

Remove stop words from remaining words after query analysis

 jerome.gasper@gmail.com authored 31 minutes ago

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>_examples</td>
<td>^ Major refactoring</td>
<td>18 days ago</td>
</tr>
<tr>
<td>_install</td>
<td>Update normalize function to replace <code>.</code> character by <code>-</code></td>
<td>16 hours ago</td>
</tr>
<tr>
<td>_scripts</td>
<td>Clean code</td>
<td>28 days ago</td>
</tr>
<tr>
<td>include</td>
<td>Remove stop words from remaining words after query analysis</td>
<td>31 minutes ago</td>
</tr>
<tr>
<td>lib</td>
<td>^ Major refactoring</td>
<td>18 days ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>[MAJOR UPDATE][DO NOT USE THIS COMMIT]</td>
<td>a month ago</td>
</tr>
<tr>
<td>.htaccess</td>
<td>Remove unused php session configuration in apache .htaccess</td>
<td>6 days ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Add codecime badge link in README</td>
<td>16 days ago</td>
</tr>
<tr>
<td>frozen ice</td>
<td>Add missing frozen ice file</td>
<td>6 months ago</td>
</tr>
</tbody>
</table>