Using WPS 1.0 standard
CNES experience review

Jérôme Gasperi | HMA AWG Meeting
Frascati - January 23rd, 2013
Testbed overview
The objective of the testbed is to develop a **WPS** interface to **OTB*** processes and to offer a complete processing chain for remote sensing images including orthorectification, cloud detection and land cover classification...accessible from a web browser

*Orfeo ToolBox (OTB) is an open source library for remote sensing image processing. The project had been initiated by the French Space Agency (CNES).*
**ORFEO Toolbox**

http://orfeo-toolbox.org/

"Simple" processes
- Binary Treshold
- Sharpening
- ...etc...

Advanced processes
- Orthorectification
- Cloud detection
- Vegetation Indice computation
- Vectorization
- Classification
- ...etc...

**Constellation**

http://www.constellation-sdi.org/

**WPS interface**

Client

Server
Web Processing Service

WPS 1.0 overview
Discover

GetCapabilities

The GetCapabilities operation provides access to general information about a live WPS implementation, and lists the operations and access methods supported by that implementation.
GetCapabilities

The GetCapabilities operation provides access to general information about a live WPS implementation, and lists the operations and access methods supported by that implementation.

DescribeProcess

The DescribeProcess operation allows WPS clients to request a full description of one or more processes that can be executed by the service. This description includes the input and output parameters and formats and can be used to automatically build a user interface to capture the parameter values to be used to execute a process.
GetCapabilities
The GetCapabilities operation provides access to general information about a live WPS implementation, and lists the operations and access methods supported by that implementation.

DescribeProcess
The DescribeProcess operation allows WPS clients to request a full description of one or more processes that can be executed by the service. This description includes the input and output parameters and formats and can be used to automatically build a user interface to capture the parameter values to be used to execute a process.

Execute
The Execute operation allows WPS clients to run a specified process implemented by a server, using the input parameter values provided and returning the output values produced. Inputs can be included directly in the Execute request, or reference web accessible resources.
<wps:Capabilities service="WPS" xml:lang="en-EN" version="1.0.0" updateSequence="1352815432361">
  <ows:ServiceIdentification>
    [...]Service information i.e. title, abstract, keywords, etc...]
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    [...]Provider information i.e. name, adress, phone, etc...]
  </ows:ServiceProvider>
  <ows:OperationsMetadata>
    [...]URLs to GetCapabilities, DescribeProcess and execute services...]
  </ows:OperationsMetadata>
  <wps:ProcessOfferings>
    <wps:Process wps:processVersion="1.0.0">
      <ows:Title>Classification</ows:Title>
      <ows:Abstract>Performs an image classification</ows:Abstract>
    </wps:Process>
    [...]
  </wps:ProcessOfferings>
  <wps:Languages>
    [...]]
  </wps:Languages>
  <wps:WSDL xlink:href=""/>
</wps:Capabilities>
  <ProcessDescription storeSupported="true" statusSupported="true" wps:processVersion="1.0.0">
    <ows:Title>Classification</ows:Title>
    <ows:Abstract>Performs an image classification</ows:Abstract>
    <DataInputs>
      [...]Standardized description of ComplexData, LiteralData or BoundingBoxData...
    </DataInputs>
    <ProcessOutputs>
      [...]Standardized description of ComplexOutput, LiteralOutput or BoundingBoxOutput...
    </ProcessOutputs>
  </ProcessDescription>
</wps:ProcessDescriptions>

storeSupported  If true, process execution is launched asynchronously - server acknowledges request and the process is performed in background. Otherwise, process execution is synchronous - server send a response back when process is finished. In this case, the HTTP connection between server and client must remains active otherwise the process result is lost.
<wps:ExecuteResponse xml:lang="en" service="WPS"
  serviceInstance="http://mon.serveur.com/wps/wps?"
  statusLocation="http://mon.serveur.com/wps/wps?request=GetExecutionStatus&executionId=5e59-6dfe"
  version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
  xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink">
  <wps:Process wps:processVersion="1.0.0">
    <ows:Title>Classification</ows:Title>
    <ows:Abstract>Performs an image classification</ows:Abstract>
  </wps:Process>
  <wps:Status creationTime="2013-01-09T16:17:08.262Z">
    <wps:ProcessAccepted>Process accepted</wps:ProcessAccepted>
  </wps:Status>
</wps:ExecuteResponse>

statusLocation The server updates the statusLocation url content according to the progress of the process. When the process is over, the status is set to <wps:ProcessSucceeded> by the server. Therefore, the Web client must regularly call the statusLocation url in order to determine the end of the process.
Testbed preliminary results
Issues

Because WPS 1.0 is not perfect ;)
Asynchronous processes management is not straightforward i.e. `storeExecute` boolean attribute; no formal `GetStatus` operation

Once launched, there is no possibility to modify the process execution (i.e. no pause, resume or abort operations)
Asynchronous processes management is not straightforward i.e. `storeExecute` boolean attribute; no formal `GetStatus` operation

Once launched, there is no possibility to modify the process execution (i.e. no pause, resume or abort operations)

This issue will be handled in WPS 2.0
Issue #2

How do i know that only Polygon is supported?

Issue #3

How do i link each polygon with the right class?

One solution is to define a Feature including both geometry and classification type...but this is quite complex to handle on client side.

Alternative solution is to define Inputs groups. This is not supported by WPS 1.0

Use case: In "assisted classification" mode, user needs to define 1...n polygons of [water, forest, cropland, etc.] as input.
Asynchronous process response document is available at `statusLocation` url. There is no way for the server to tell the client how long this url will be valid (hours, months, years, etc. ?) - and no way to know how long the result url within the response document will be valid too...
WPS 1.0 specification is simple and readable!
WPS 1.0 specification is **simple** and **readable**!

**Easy** to implement on server. A bit more tricky on the client side due to inputs genericity (e.g. GML features)
WPS 1.0 specification is **simple** and **readable**!

**Easy** to implement on server. A bit more tricky on the client side due to inputs genericity (e.g. GML features)

It is a Web service technology so it is naturally **made for the cloud**
WPS 1.0 specification is **simple** and **readable**!

**Easy** to implement on server. A bit more tricky on the client side due to inputs genericity (e.g. GML features)

It is a Web service technology so it is naturally **made for the cloud**

Limitations with asynchronous processes in WPS 1.0 will be handled in WPS 2.0
 jerome.gasperi@cnes.fr