INTRODUCTION TO WEB PROCESSING SERVICES

Benjamin Pross, Christoph Stasch
52°North GmbH
Geospatial Sensor Web Conference, 2018-09-03
OVERVIEW

• Web-based Geoprocessing
  • Why and how?
  • OGC WPS

• Implementations & Details about the 52°North WPS

• Example applications
MAIN FOCUS OF 52°NORTH

Sensor Web Enablement

Geoprocessing

Desktop Apps

Web Apps

SDIs, SOA, Big Data

User icons made by Freepik from www.flaticon.com are licensed as CC BY 3.0
Problems: Why Web-based Geoprocessing and WPS?
GEOPROCESSING

Raw data → Processing → Value-added data products → Processing → Information products
GEOPROCESSING – EARLIER APPROACH

Desktop GIS

Input data

Output data
**WEB SERVICE - APPROACH**

- Web Applications
- Desktop GIS
- ShakeMap Service
- Data Service
- Other Web Services

- ShakeMap Computation
- Risk Service
  - Exposure, Vulnerability Service

Standardized APIs + Data Formats
OGC Web Processing Service (WPS)

- **WMS** – Maps as Images (jpg, tiff, ...)
- **WCS** – Coverages (geoTiff, netCDF, ...)
- **WFS** – Vector Data (GML, shp)
- **SOS** – Observations (O&M, SweCommon, ...)
**GEOPROCESSING IN THE WEB, BECAUSE...**

...the set-up of software is complicated

...processing is tightly coupled to data that is available only remotely

...I want to reuse processing in different environments

...I lack the computational capacity in my own environment

...I want to couple different technologies („Processing Mash-Ups“).
...the analysis should be automated and controllable via the web

Coordinate transformation

Quality assurance

Data fusion
OVERVIEW: WHAT IS THE OGC WPS?
OGC Web Processing Service - Overview

- Official OGC Standard since 2007, Version 2.0.0 since 2015:
  - [http://www.opengeospatial.org/standards/wps](http://www.opengeospatial.org/standards/wps)

- Standardized description of geoprocessing functionality („processes“)
  - Identifier
  - Textual description
  - Input and output parameters

- Predefined service operations for the description and execution of processes
  (synchronous, asynchronous)
  - DescribeProcess, Execute, GetResult

- Software:
  - 52°North WPS, PyWPS, Zoo WPS, ArcGIS Server, ERDAS Imagine, …
**OGC Web Processing Service – Basic Operations**

**Clients**
- Desktop Apps
- Web Apps
- SDIs, SOA, Cloud Environments

**WPS**
- Process 1
- Process 2
- Process n

- GetCapabilities
  - List of processes
- DescribeProcess
  - Detailed process description
- Execute
  - Processing result
WPS – Which processes are available?

Clients
- Desktop Apps
- Web Apps
- SDIs, SOA, Big Data

GetCapabilities

List of processes

WPS
- Process 1
- SimpleBufferAlgorithm
- Process n
WPS – GetCapabilities Operation

• Request via URL (HTTP GET with Key-Value-Pair encoding) or XML Request (HTTP POST)
• Returns service description of the WPS
• Basic information:
  • Endpoints
  • Technical request mechanisms
  • Information about the service provider/access constraints
• Short information about the offered processes
**WPS – What does the process do? What are the inputs/outputs?**

**Clients**
- Desktop Apps
- Web Apps
- SDIs, SOA, Big Data

**WPS**
- GetCapabilities: List of processes
- DescribeProcess: Detailed process description

**Process 1**
- SimpleBufferAlgorithm

**Process n**
WPS: **Describe Process Operation**

- Request via URL (HTTP GET with Key-Value-Pair encoding) or XML Request (HTTP POST)
- Returns detailed description of a process based on the process id
- Defines the inputs and outputs
  - IDs
  - Default data formats
  - Further supported data formats
**WPS – How do I execute a process?**

**Clients**
- Desktop Apps
- Web Apps
- SDIs, SOA, Big Data

**WPS**
- GetCapabilities
  - List of processes
- DescribeProcess
  - Detailed process description
- Execute
  - Processing result

- Process 1: SimpleBufferAlgorithm
- Process n
WPS – Execute Operation

• XML Request (HTTP POST)
• Execution of an offered process
• Request:
  • Must contain id and input parameters according to the process description
• Result
  • Can be returned directly or as reference to a web accessible resource
• Can be executed asynchronously for long running processes -> Client doesn’t directly get the result, but can request the status of the execution
**WPS – How to Transfer the Input Data**

**Option 1: Direct transfer**

**Option 2: Reference to web accessible resource**

**Option 3: Query on encapsulated data**
ASYNCHRONOUS EXECUTION: HOW DO I EXECUTE LONG RUNNING PROCESSES
WPS – SYNCHRONOUS EXECUTION

Client

Send request

Waits for response

Receives request

Processes result

Sends result

WPS

Receive result and uses it

Send request

Waits for response

Receives request

Processes result

Sends result
**WPS – ASYNCHRONOUS EXECUTION (PUSH-MODEL)**

Sequence Diagram

- **Client**
  - Sends request
  - Does something different
  - Receives the result and uses it

- **WPS**
  - Receives request
  - Processes result
  - Sends result
**WPS Spec 2.0 – Asynchronous Execution**

**Clients**
- Desktop Apps
- Web Apps
- SDIs, SOA, Big Data

**WPS**
- Process 1
- SimpleBufferAlgorithm
- Process n

- Execute(mode=async,...)
  - JobID, StatusInfo
- GetStatus(JobID)
  - StatusInfo
- GetResult(JobID)
  - Result
EXAMPLE APPLICATIONS: WHERE ARE WPS USED?
USGS

Dataset Selection

4km Monthly Parameter-elevation Regressions on Independent Slopes Model Monthly Climate Data for the Continental United States.

Abstract
This dataset was created using the PISM (Parameter-elevation Regressions on Independent Slopes Model).

Bias Corrected Constructed Analogs V2 Daily Climate Projections

Abstract
This archive contains projections of daily BCC/CMIP5 and CMIP5 projections of precipitation, daily ...

Bias Corrected Spatially Downscaled Monthly CMIP5 Climate Projections

Abstract
This archive contains 234 projections of monthly GCM/CMIP5 projections of precipitation and monthly...

Bias Corrected Spatially Downscaled Monthly Climate Projections

Abstract
This archive contains fine spatial-resolution translations of 112 contemporary climate projections ...
EXAMPLE APPLICATION: TAMIS
Current WPS topics
REST API

• Currently in the process of standardization
• Based on a proposal in the OGC Testbed 12 - REST Architecture Engineering Report (OGC 16-035)
• Implementation as proxy for „normal“ WPSs available

REST proxy implementation: https://github.com/52North/wps-proxy
Engineering Report: http://docs.opengeospatial.org/per/16-035.html
INTERFACE FOR CLOUD-BASED GEPREPROCESSING

Quelle: http://www.opengeospatial.org/node/2526#Cloud
HOSTED PROCESSING

Clients
- Desktop Apps
- Web Apps
- SDIs, SOA, Cloud Environments

DeployProcess(DockerImage)

Process 1
- ID of the new process
- Detailed process description

DescribeProcess

Process 2
- Execute
- Processing result

WPS
- Docker
- Process n
Interoperable Geoprocessing Workflows

• BPMN/Description Languages
• Provenance/Metadata (Discovery)
• Workflow Validation
• Uncertainty Propagation
• Security
• Granularity (Microservices)
Server Implementations: How can I provide geoprocessing functionality in a WPS?
Server Implementations

• OGC lists 54 implementations
• Commercial:
  • ESRI, FME, Intergraph, Envitia, ERDAS, …
• Open Source:
  • 52°North, pyWPS, ZOO, Geoserver, …
52°NORTH WPS

• Version 3.3.x → Beta-Release of version 4.0.0 available

• Supports (all) features and operations of the WPS specification Version 1.0.0 and 2.0

• Support of:
  • GRASS 7, Sextante, R, Java/Python

• Java Process/R Script upload via Web UI

• Parser/Generators for common data formats:
  • SHP-Files, GML, GeoJSON, GeoTIFF, NetCDF, …
52° North WPS Admin App

Documentation
- To learn more about the specification visit the OGC website.
- To learn more about this implementation visit the 52° North Geoprocessing Community website.

Administration
Please login to access administration pages.

Examples
Requests
- GetCapabilities request using HTTP GET

Test Client
Open the test client of this WPS instance here: 52°North WPS test client.
52°NORTH WPS DEPLOYMENT PATTERNS

• Extending an existing WPS
• Deployment of annotated scripts (currently R)
• Custom Binaries with configuration files
• Databases
• Coupling with Geoserver
WPS EXTENSION

• Direct extension of a existing WPS Server implementation
  • Reuse of I/O handlers
• Example: Implementation of an algorithm in Java as direct extension of the 52N WPS
Execution of Custom Binaries

- Execution of a program in binary code
- Configuration via config files
  - Parameters for controlling the process
  - Inputs/Outputs are stored locally and are referenced using config parameters
WPS as Rich Data Interface

- WPS as interface for a large data store
- Dynamic views can be generated using input parameters
  - Inputs don’t contain data for processing
Deployment of R Scripts

- Annotation in R scripts define inputs/outputs
- Upload of annotated R scripts allows deployment as WPS processes

```r
# vps.de: id = Random, title = Random number generator,
# abstract = Generates random numbers for uniform distribution:
# vps.in: min, double; Minimum, All outcomes are larger than min, value = 0;
# vps.in: max, double; Maximum, All outcomes are smaller than max, value = 1;
# vps.in: n, integer, Amount of random numbers, value = 100
# random number:
# x = runif(n, min=0, max=max)
output = "outputfilename"
write.Cable(x, output)
# vps.out: output, text; Random number list, Textfile containing n random numbers in one column
```
ArcGIS Server WPS Extension

CLIENTS:
HOW CAN I USE GEOPROCESSING FUNCTIONALITY OF A WPS?
Client Implementations

- 52°North ArcGIS WPS Client
  - ArcGIS Extension in collaboration with ESRI Inc.
  - Available as Open Source
- 52°North WPS-JS
  - JavaScript library for the creation of Web clients
- Quantum-GIS WPS – Plugin
- Custom clients for specific applications, e.g. USGS
52°NORTH WPS-JS-CLIENT

- **wps-js:**
  - JavaScript client library

- **Wps-js-client:**
  - JavaScript client
  - Based on the Angular Framework
Adding of a WPS
52° North ArcGIS WPS Client (II)

Toolbox for processes

Input form for the execution
52°NORTH ArcGIS WPS Client (III)

Integration in the Model Builder

Result map
**SUMMARY**

- OGC WPS as standardized service for the access to geoprocessing functionality in the Web
  - Complementary to OGC – data services
  - Asynchronous execution
- REST Binding currently in standardization process:
  - Leaner and easier to use than the SOAP Binding
- 52N REST Proxy already usable with existing WPS servers
  - Currently restricted, e.g. the JSON Encoding is currently 52n-specific (not yet standardized)
THANKS!

Benjamin Pross  
WPS Expert  
b.pross@52north.org

Christoph Stasch  
Geodata processing  
c.stasch@52north.org

52N WPS Website: [https://52north.org/software/software-projects/wps/](https://52north.org/software/software-projects/wps/)

52N WPS Mailinglist: [https://list.52north.org/mailman/listinfo/geoprocessing](https://list.52north.org/mailman/listinfo/geoprocessing)
SHORT DEMO
ANNEX: WPS REST EXTENSION
**Motivation For a REST Extension**

- **WPS 2.0 and other OGC specifications normally define a SOAP Binding as necessary interface**
  - Exceptions: OGC Web Map Tiling Server and Sensor Things API

- **REST APIs with JSON Encodings leaner for implementations of Web clients and easier to use**
  - Focus on resources (not on operations)
  - Use of standard HTTP operations for CRUD of resources

- **In OGC Testbed 12 REST APIs for different OGC services were tested and described in the Testbed 12 REST Architecture Engineering Report**

http://docs.opengeospatial.org/per/16-035.html
EXAMPLE: WEB FEATURE SERVICE 3.0 (DRAFT)

https://github.com/opengeospatial/WFS_FES
EXISTING GEOPROCESSING REST APIs

• APIs for existing GIS software
  • ESRI’s Geoservices REST API
  • FME Server
  • Hexagon Foundation API
  • 

• APIs for particular geoprocessing applications
  • Geocoding (Bing Maps, Google, Yahoo, …)
  • Routing (Google, HERE Maps, …)
  • …
WPS Resource Model

- Based upon the general process model of WPS 2.0
- Hypermedia approach
  - Capabilities contains link to ProcessCollection
  - ProcessCollection contains link to single Process –ressources
  - ...
# Endpoints for Resource Access

<table>
<thead>
<tr>
<th>Ressource</th>
<th>Beschreibung</th>
<th>HTTP Operation</th>
<th>Endpunkt</th>
<th>Message Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capabilities</td>
<td>Request of the service description</td>
<td>HTTP GET</td>
<td>{WPSRestBaseURL}</td>
<td>-</td>
</tr>
<tr>
<td>ProcessCollection</td>
<td>Request of the list of processes</td>
<td>HTTP GET</td>
<td>{WPSRestBaseURL}/processes</td>
<td>-</td>
</tr>
<tr>
<td>Process</td>
<td>Request of a single process description</td>
<td>HTTP GET</td>
<td>{WPSRestBaseURL}/processes/{processID}</td>
<td>-</td>
</tr>
<tr>
<td>JobCollection</td>
<td>Request of the list of jobs (executions) of a process</td>
<td>HTTP GET</td>
<td>{WPSRestBaseURL}/processes/{processID}/jobs</td>
<td>-</td>
</tr>
<tr>
<td>Job</td>
<td>Execution of a process/creation of a new job</td>
<td>HTTP POST</td>
<td>{WPSRestBaseURL}/processes/{processID}/jobs</td>
<td>Execute Request in JSON</td>
</tr>
<tr>
<td>Job</td>
<td>Request of the status of a job</td>
<td>HTTP GET</td>
<td>{WPSRestBaseURL}/processes/{processID}/jobs/{jobID}</td>
<td>-</td>
</tr>
<tr>
<td>Outputs</td>
<td>Request of the results of a job</td>
<td>HTTP GET</td>
<td>{WPSRestBaseURL}/processes/{processID}/jobs/{jobID}/results</td>
<td>-</td>
</tr>
</tbody>
</table>
52N WPS-REST-Proxy

• Implements REST extension as proxy for existing WPS instances
  • Advantage: Can be used together with existing WPS 2.0 servers
  • Disadvantage: Increased communication effort

• GitHub Repo:
  • https://github.com/52North/wps-proxy

• Example instance for testing:
  • http://geoprocessing.demo.52north.org:8080/wps-proxy

• Tutorial:
  • https://wiki.52north.org/Geoprocessing/WPSRESTProxy
**REQUEST CAPABILITIES**

HTTP GET `<baseURL>`

- No predefined pattern for endpoint-URL
- Returns Capabilities document containing the list of available processes

```json
1 {
  "Capabilities": {
    "ServiceIdentification": {
      "Title": "52\(^{°}\) North WPS 4.0.0-SNAPSHOT",
      "Abstract": "Service based on the 52\(^{°}\)North implementation of WPS 1.0.0",
      "ServiceType": "WPS",
      "ServiceTypeVersion": ["1.0.0", "2.0.0"],
      "Fees": "NONE",
      "AccessConstraints": "NONE"
    },
    "ServiceProvider": {
      "ProviderName": "52\(^{°}\)North",
      ...
    }
  },
  "Contents": {
    "ProcessSummaries": [{
      "Identifier": "testbed12.fo.DouglasPeuckerAlgorithm",
      "title": "testbed12.fo.DouglasPeuckerAlgorithm",
      "processVersion": "1.0.0",
      "jobControlOptions": "sync\-execute",
      "outputTransmission": "value",
      "url": "http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/org.n52.wps.server.algorithmdouglaspeuckeralgorithm"
    },...
  ],
  "service": "WPS",
  "version": "2.0.0"
}
```

http://geoprocessing.demo.52north.org:8080/wps-proxy
**Request Process List**

HTTP GET `<baseUrl>/processes`

- Returns a list of short process summaries containing links to detailed process descriptions

```json
1 {  
2   "ProcessSummaries": [  
3     {  
4       "identifier": "testbed12.fo.DouglasPeuckerAlgorithm",  
5       "title": "testbed12.fo.DouglasPeuckerAlgorithm",  
6       "_processVersion": "1.0.0",  
7       "_jobControlOptions": "sync-execute",  
8       "outputTransmission": "value",  
9       "url": "http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/n52.wps.server.algorithm.JTSConvexNullAlgorithm"  
10 },  
11 ]  
12 }
```

Link to detailed process description

http://geoprocessing.demo.52north.org:8080/wps-proxy/processes
**Request Process Description**

HTTP GET `<baseURL>/processes/<process-id>`

- Returns the detailed description of a process including input and output parameters

```json
{
  "ProcessOffering": {
    "Process": {
      "Title": "org.n52.wps.server.algorithm.JTSConvexHullAlgorithm",
      "Identifier": "org.n52.wps.server.algorithm.JTSConvexHullAlgorithm",
      "Input": [
        {
          "Title": "data",
          "Identifier": "data",
          "ComplexData": {
            "Format": [
              {
                "_default": "true",
                "_mimeType": "application/vnd.geosap-json"
              }
            ],
            "_minOccurs": "1",
            "_maxOccurs": "1"
          }
        }
      ],
      "Output": [
        {
          "Title": "result",
          "Identifier": "result",
          "ComplexData": {
            "Format": [
              {
                "_default": "false",
                "_mimeType": "application/vnd.geosap-json"
              }
            }
          }
        }
      ],
      "processVersion": "1.1.0",
      "_jobControlOptions": "sync-execute async-execute",
      "_jobIdentifier": "uniqueidentifier",
      "execute-url": "http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/org.n52.wps.server.algorithm.JTSConvexHullAlgorithm/jobs"
    }
  }
}
```
**Execution of a Process (1)**

HTTP POST

`<baseURL>/processes/<process-id>/jobs`

- **Parameter:**
  - Execute Request in JSON (see to the right)

- **Optional URL parameter:**
  `sync-execute=true|false` *(default false)*

- A new job resource (process execution) is created

```json
{
    "Execute": {
        "Identifier": "org.n52.wps.server.algorithm.JTSConvexHullAlgorithm",
        "Input": [  
            {  
                "ComplexData": {  
                    "_mimeType": "application/wkt",
                    "_text": "POLYGON((847666.55940585 6793166.084248,
                    ...,  
                    847666.55940585 6793166.084248))"
                },  
                "_id": "data"
            }
        ],
        "output": [{  
            "_mimeType": "application/wkt",
            "_id": "result",
            "_transmission": "value"
        }],  
        "_service": "WPS",
        "_version": "2.0.0"
    }
}
```

HTTP: `http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/org.n52.wps.server.algorithm.JTSConvexHullAlgorithm/jobs`
**Execution of a Process (II)**

- **Asynchronous** execution (Examples to the right):
  - HTTP 201 with link to job resource
  - After execution is finished link to results

```json
1  { 
2    "StatusInfo": { 
3      "JobID": "c731d14b-1de6-499c-9317-20224e056012", 
4      "Status": "Running", 
5      "Progress": 0 
6    } 
7  }
```

- **Synchronous** execution (Example on next slide):
  - JSON result document (success|failure)

```json
1  { 
2    "StatusInfo": { 
3      "JobID": "c731d14b-1de6-499c-9317-20224e056012", 
4      "Status": "Succeeded", 
5      "Output": "http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/org.n52.wps.server.algorithm.3DConvexHullAlgorithm/jobs/c731d14b-1de6-499c-9317-20224e056012/output" 
6    } 
7  }
```

Running job

```
```

Finished job with link to results
REQUEST PROCESSING RESULTS

HTTP GET:

```
<baseURL>/processes/<process-id>/
jobs/<job-id>/outputs
```

• Returns JSON result document
WPS Testclient verfügbar unter: http://geoprocessing.demo.52north.org:8080/wps/test_client