52north
exploring horizons

Introduction to Web Processing Services

Benjamin Pross, Christoph Stasch
52°North GmbH

Geospatial Sensor Web Conference, 2018-09-03

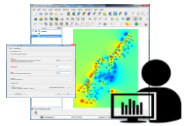
Overview

- 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

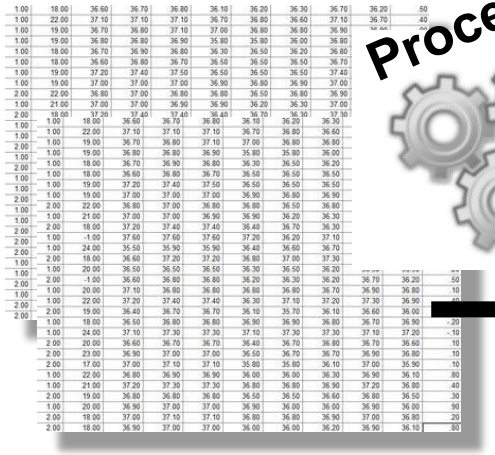
SDIs, SOA, Big Data

52north
exploring horizons

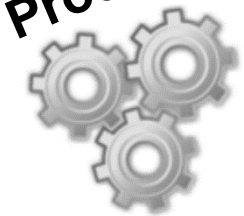
Problem: Why Web-based Geoprocessing and WPS?

GEOPROCESSING

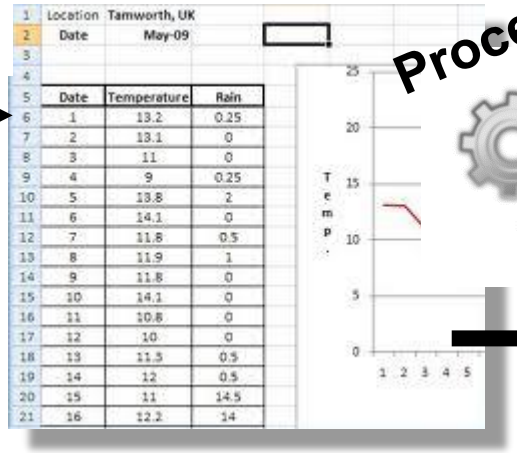
Raw data



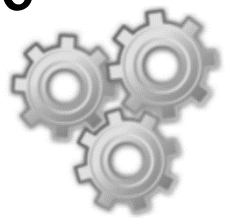
Processing



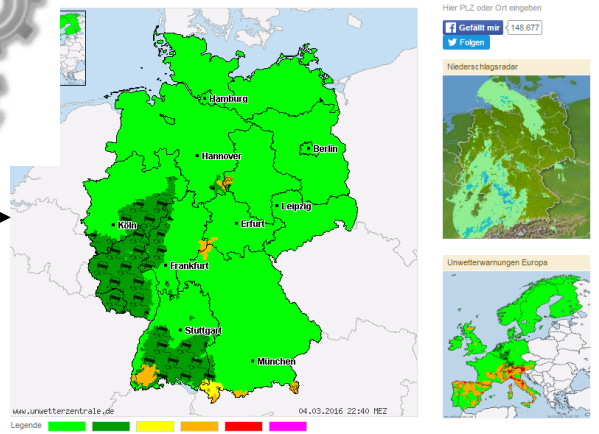
Value-added data products



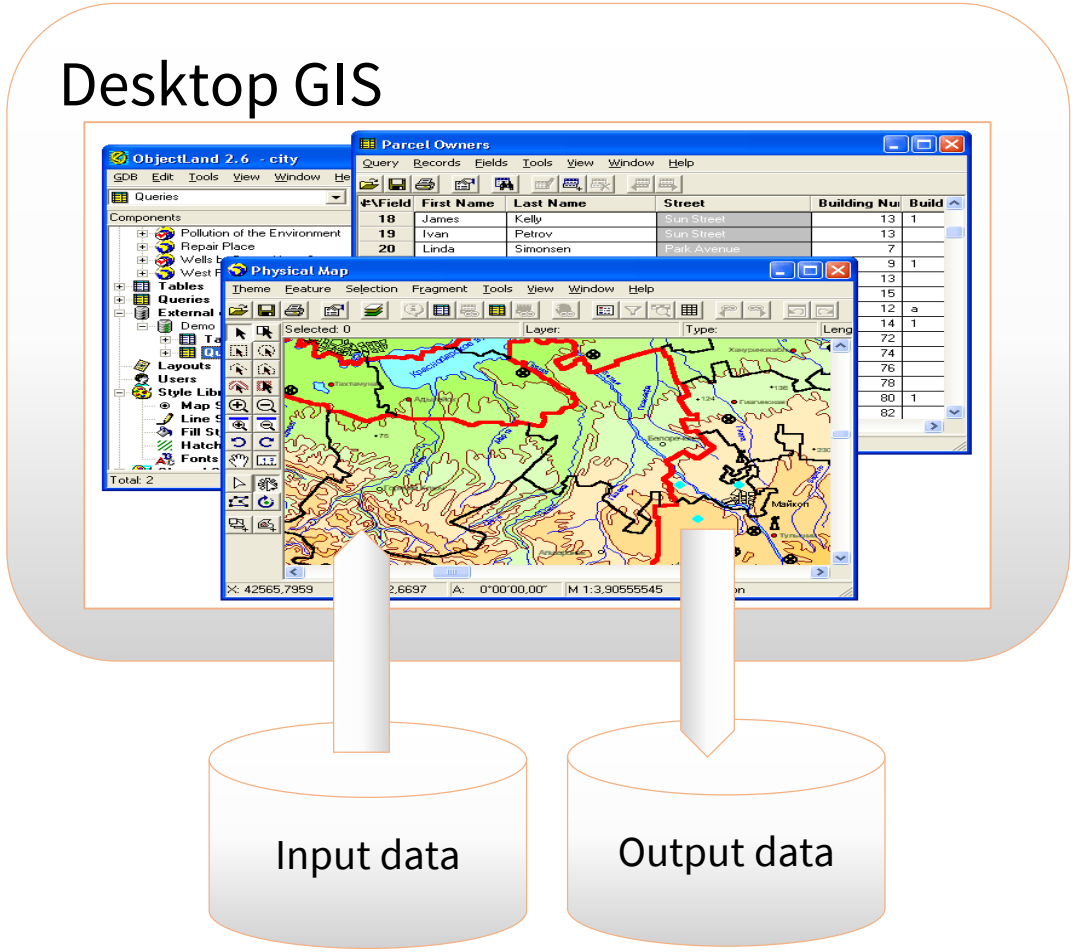
Processing



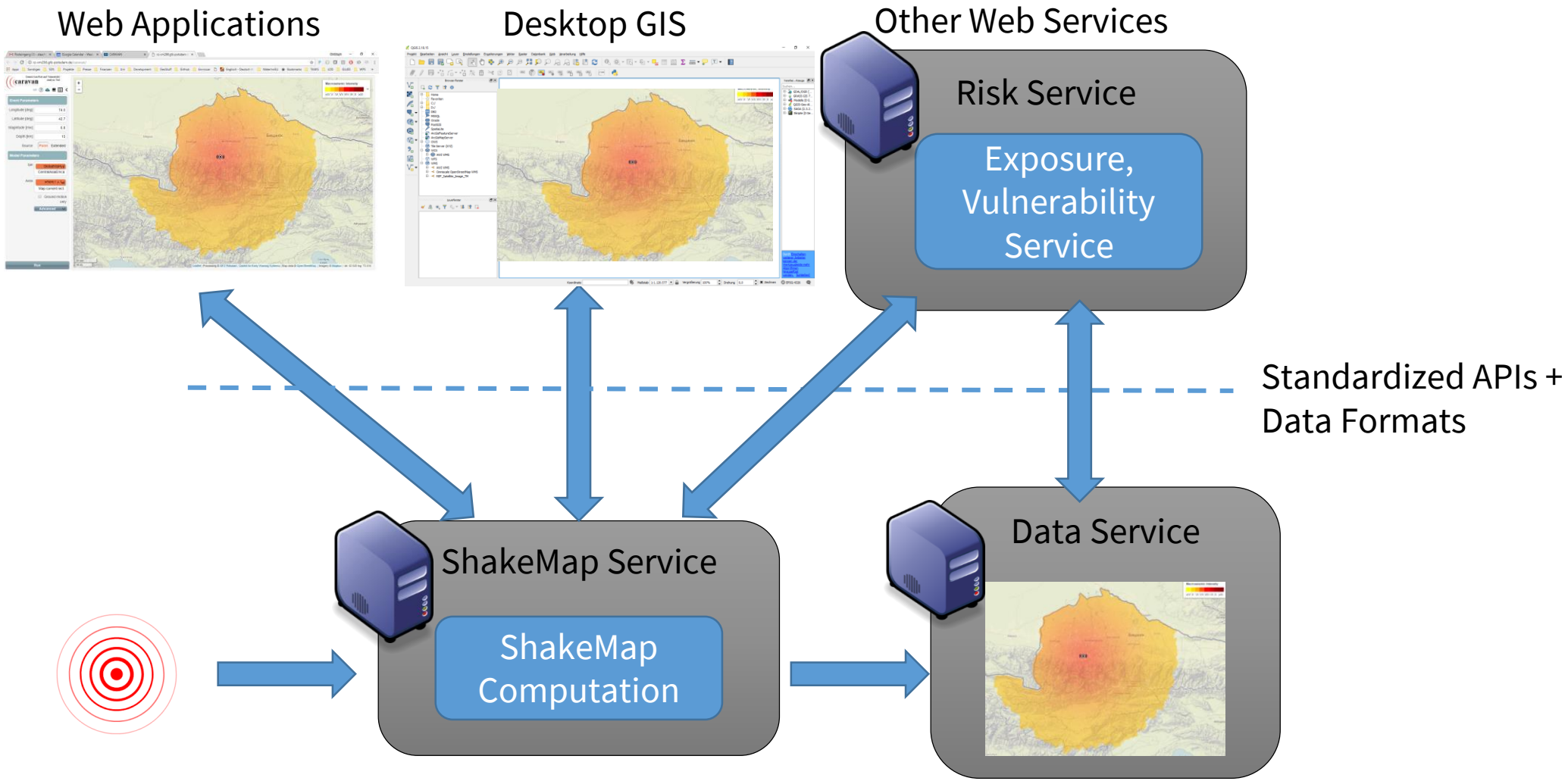
Information products



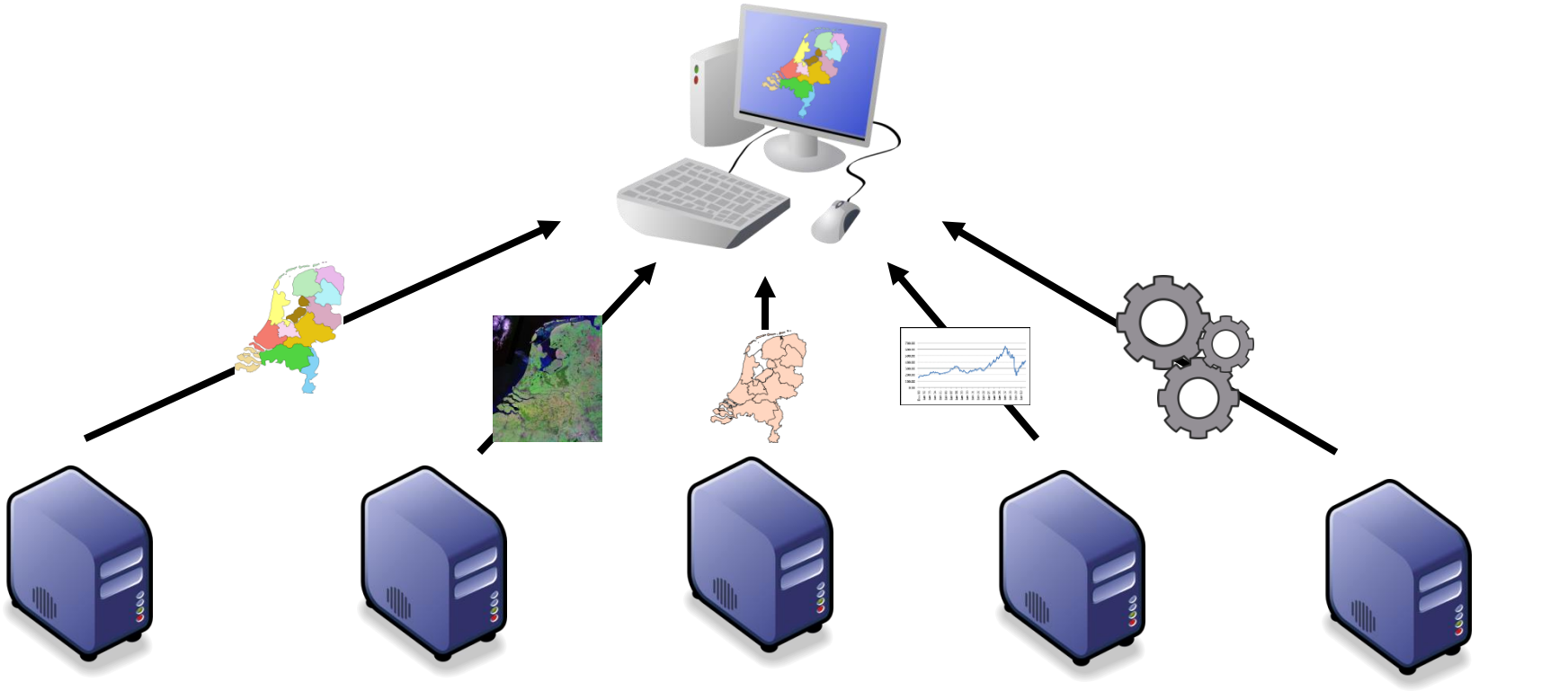
Geoprocessing – Earlier Approach



Web Service - Approach



OGC Web Processing Service (WPS)



WMS – Maps as Images (jpg, tiff, …)

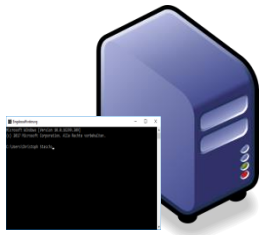
WCS – Coverages (geoTiff, netCDF, …)

WFS – Vector Data (GML, shp)

SOS – Observations (O&M, SweCommon,…)

WPS – Geoprocesses, Simulations, …

Geoprocessing in the Web, because...

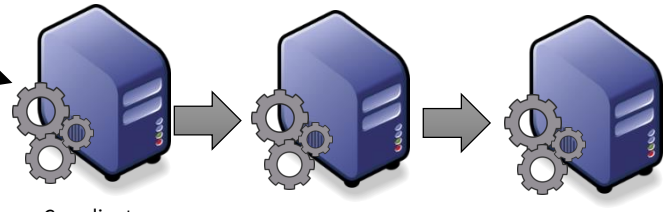


…the set-up of software is complicated

…I lack the computational capacity in my own environment

…I want to reuse processing in different environments

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



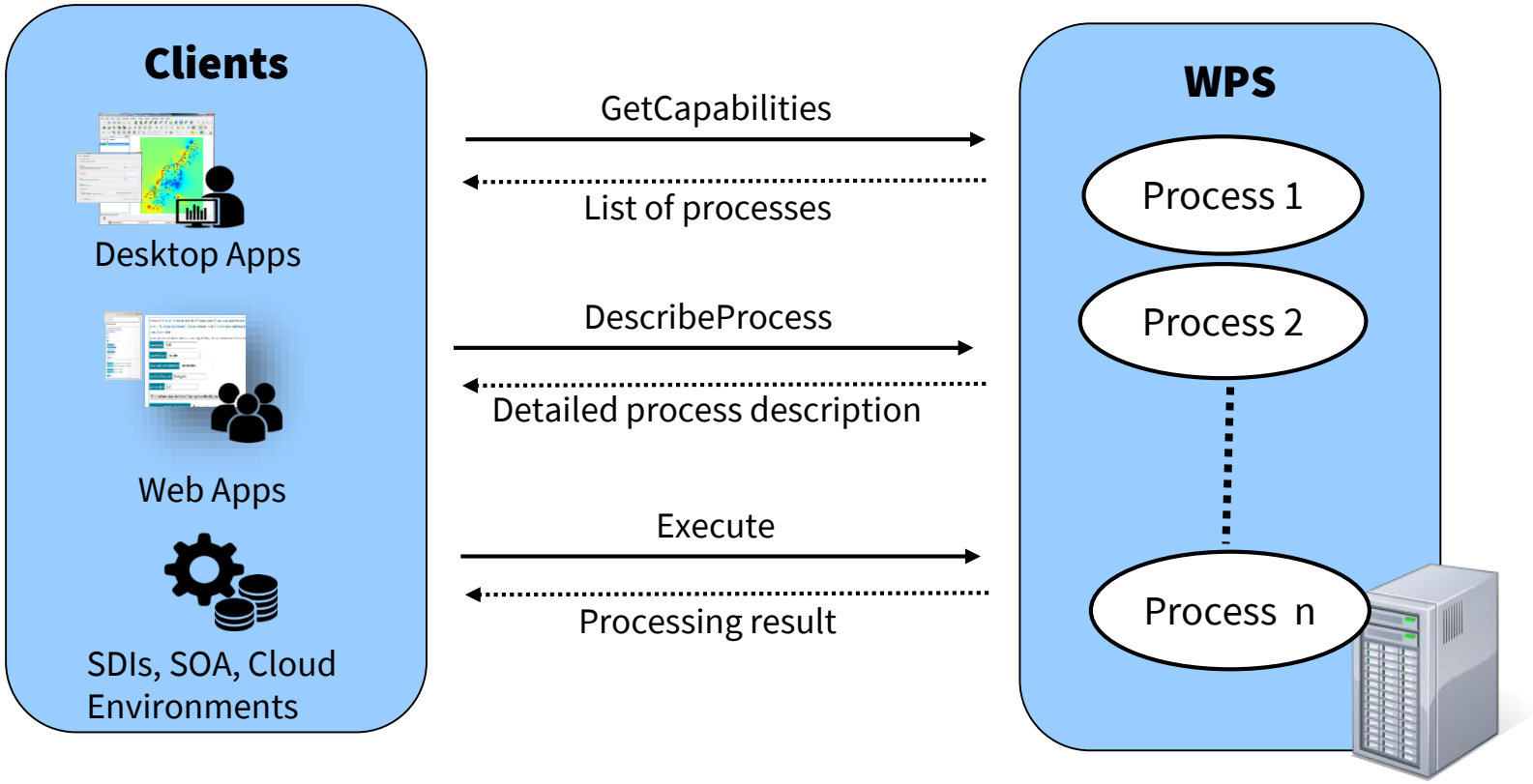
…I want to couple different technologies („Processing Mash-Ups“).
…the analysis should be automated and controllable via the web

Overview: What is the OGC WPS?

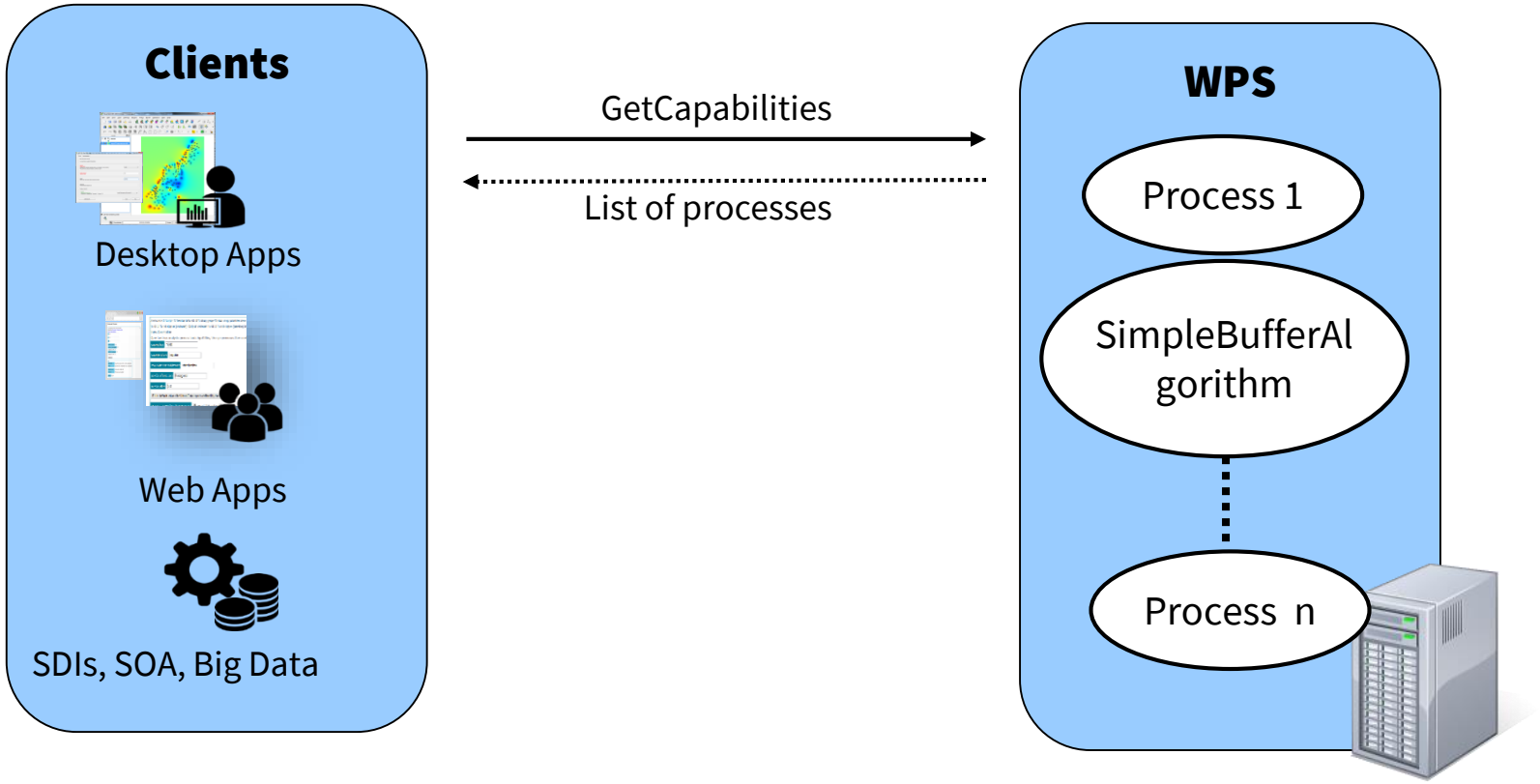
OGC Web Processing Service - Overview

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



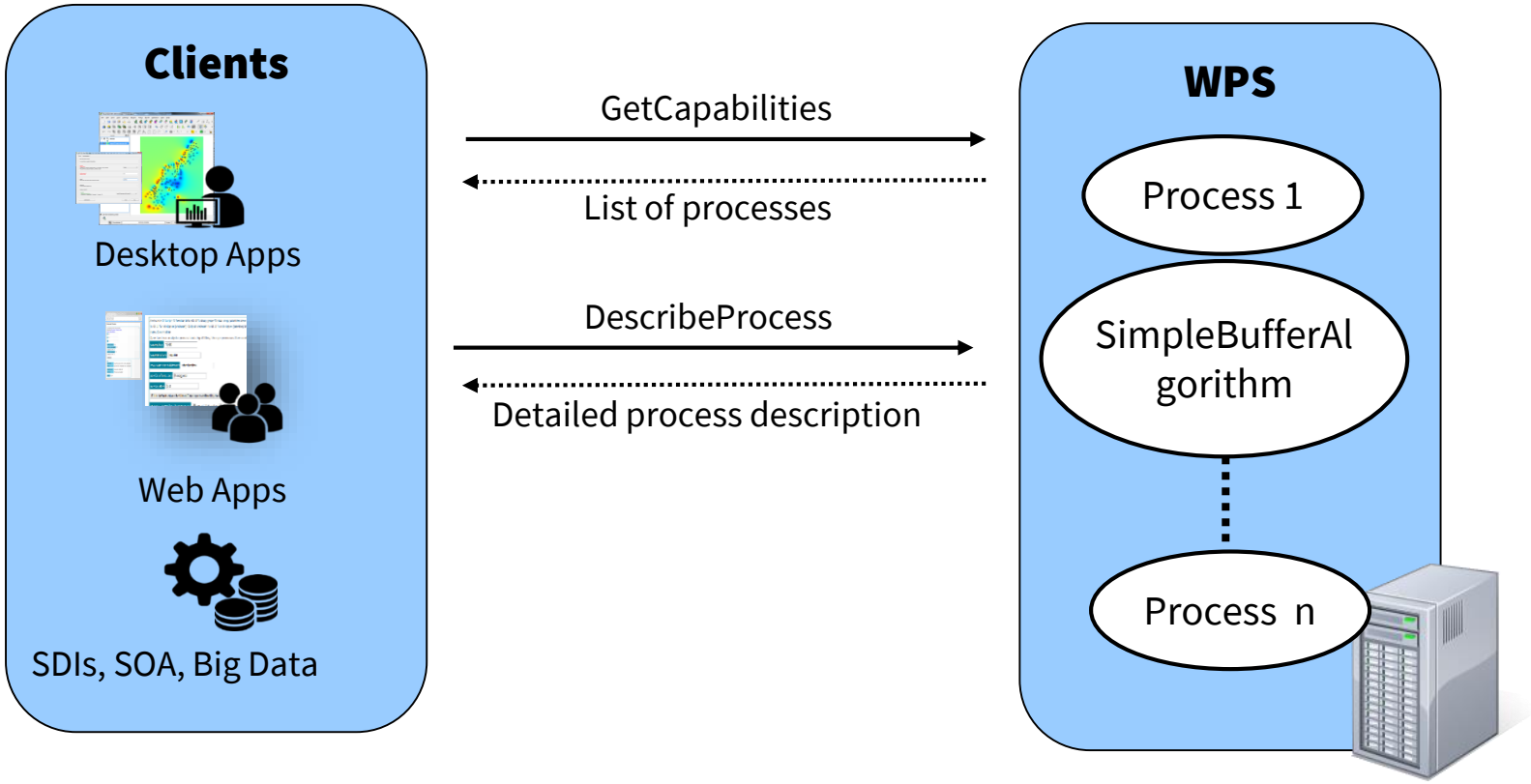
WPS – WHICH PROCESSES ARE AVAILABLE?



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

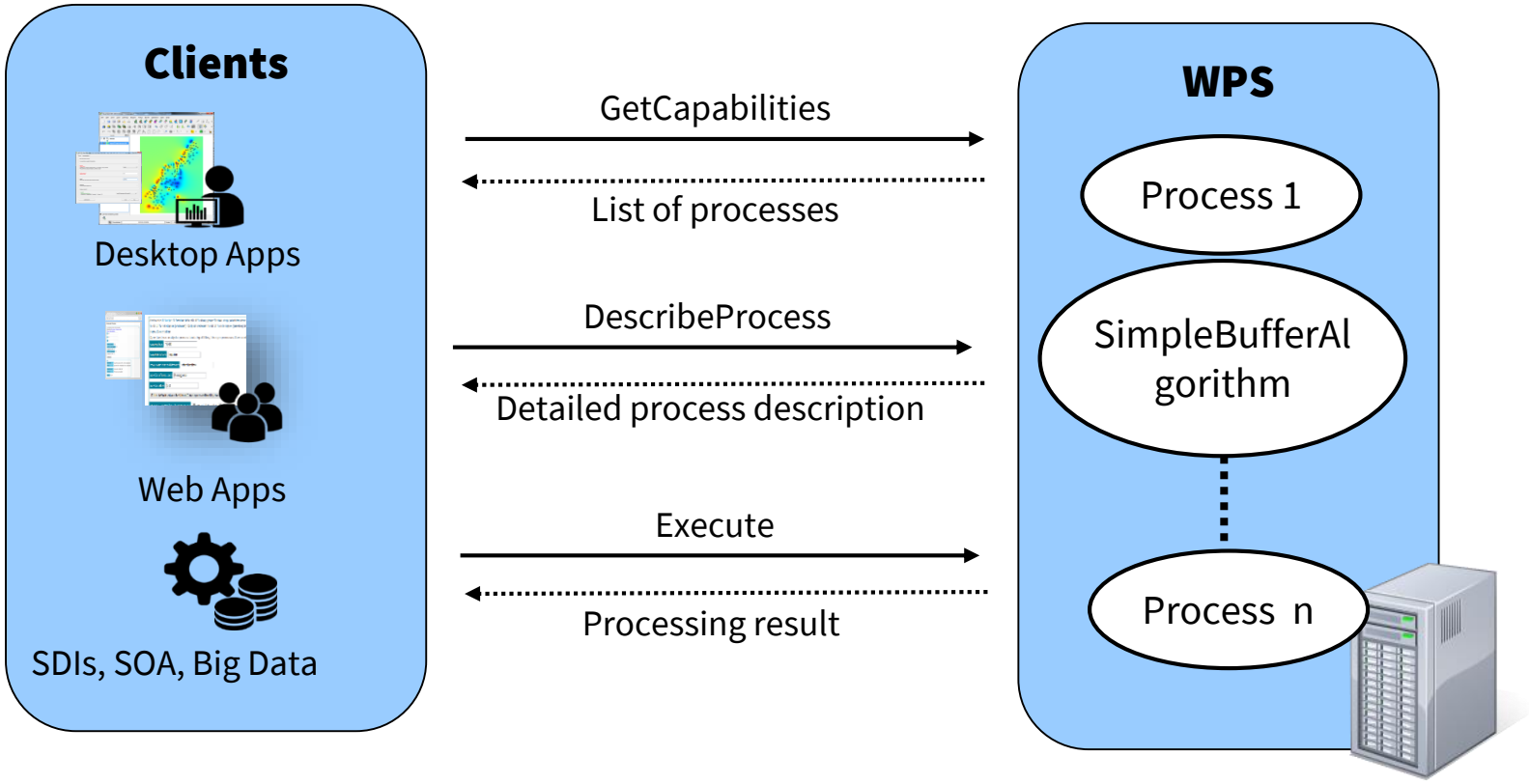
WPS – What does the process do? What are the inputs/outputs?



WPS: DescribeProcess 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

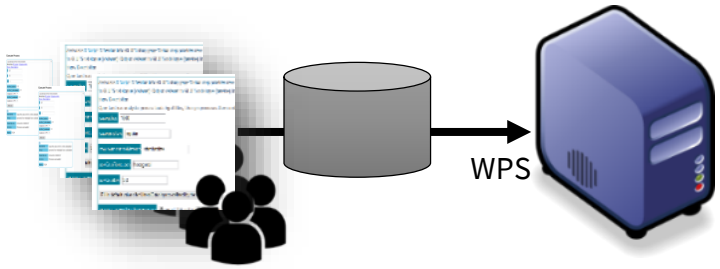
WPS – How do I execute a process?



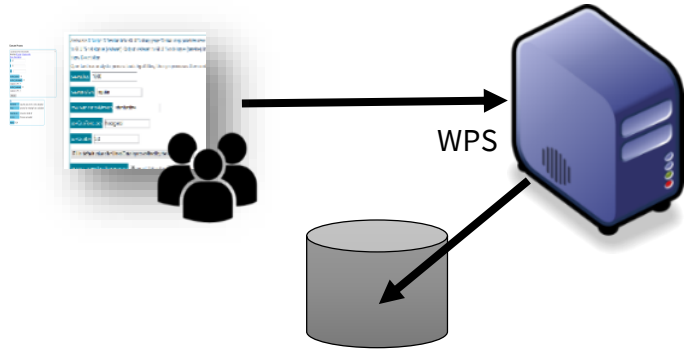
WPS – Execute Operation

- 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

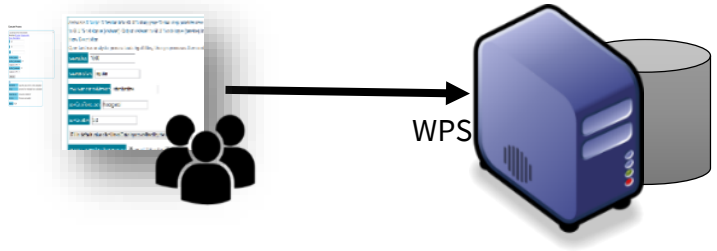
WPS – How to transfer the input data



Option 1: Direct transfer



Option 2: Reference to web accessible resource

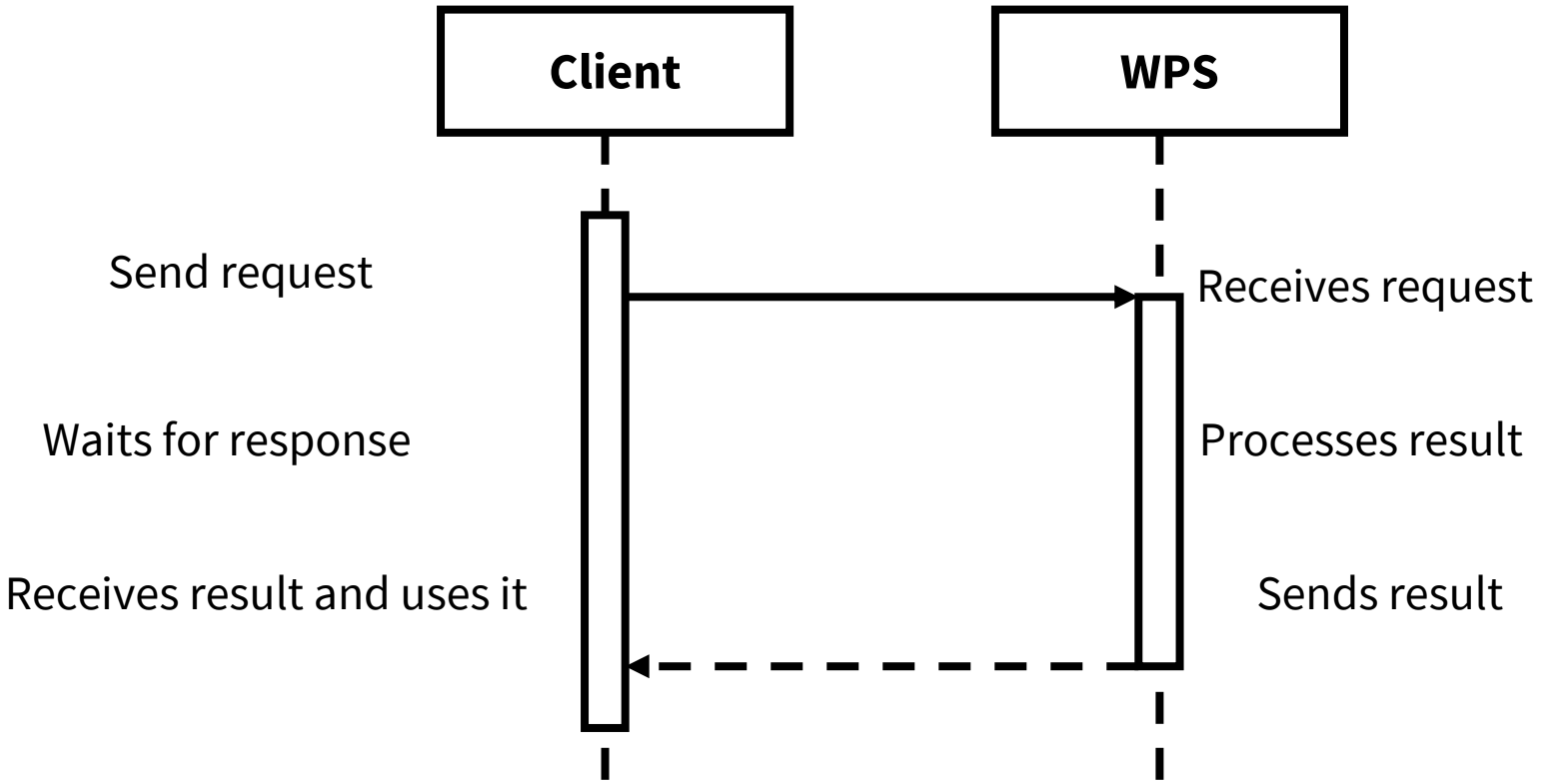


Option 3: Query on encapsulated data

52north
exploring horizons

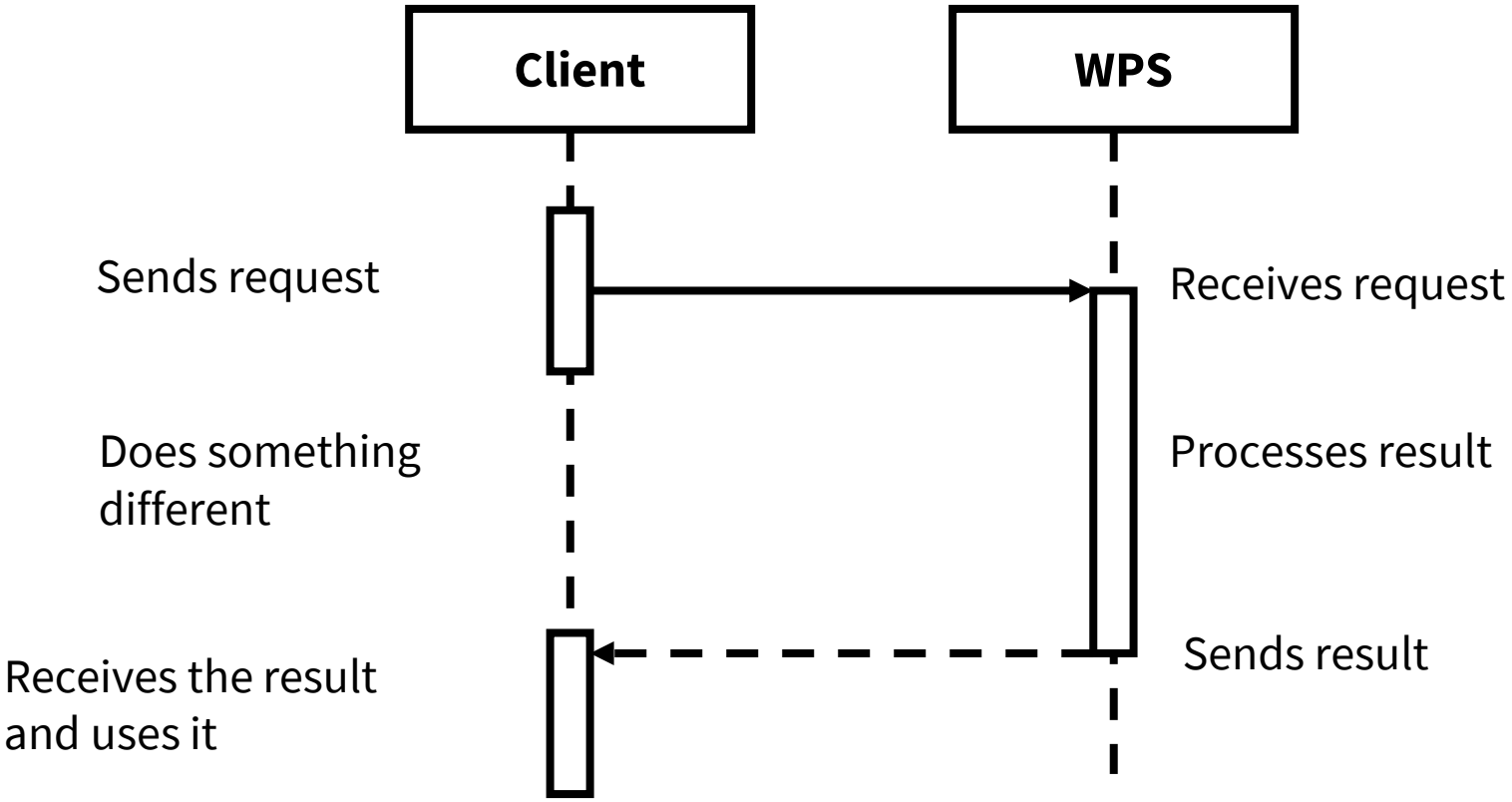
Asynchronous execution: How do I execute long running processes

WPS – Synchronous execution

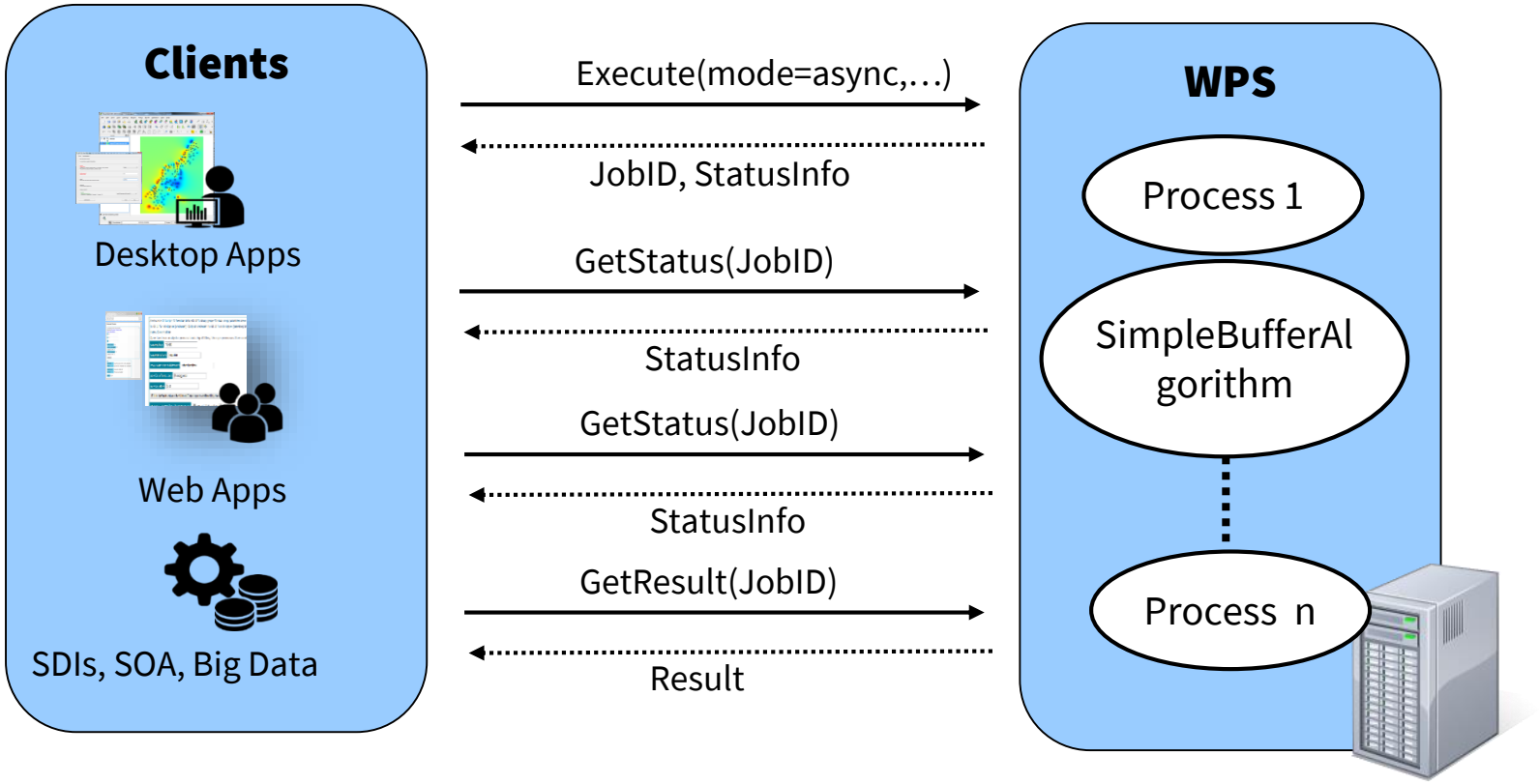


WPS – Asynchronous Execution(Push-Model)

Sequence Diagram

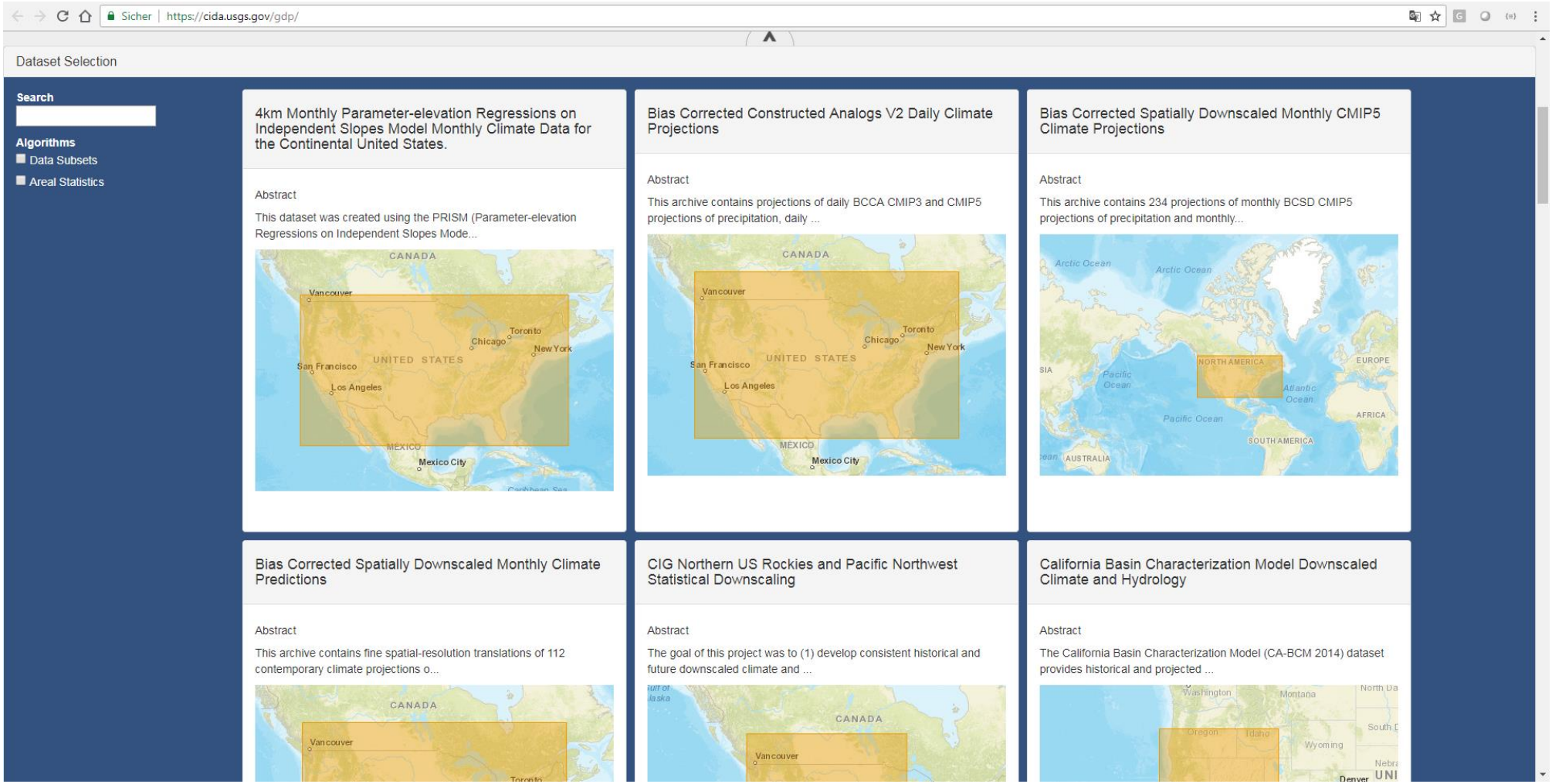


WPS Spec 2.0 – Asynchronous execution



Example applications: Where are WPS used?

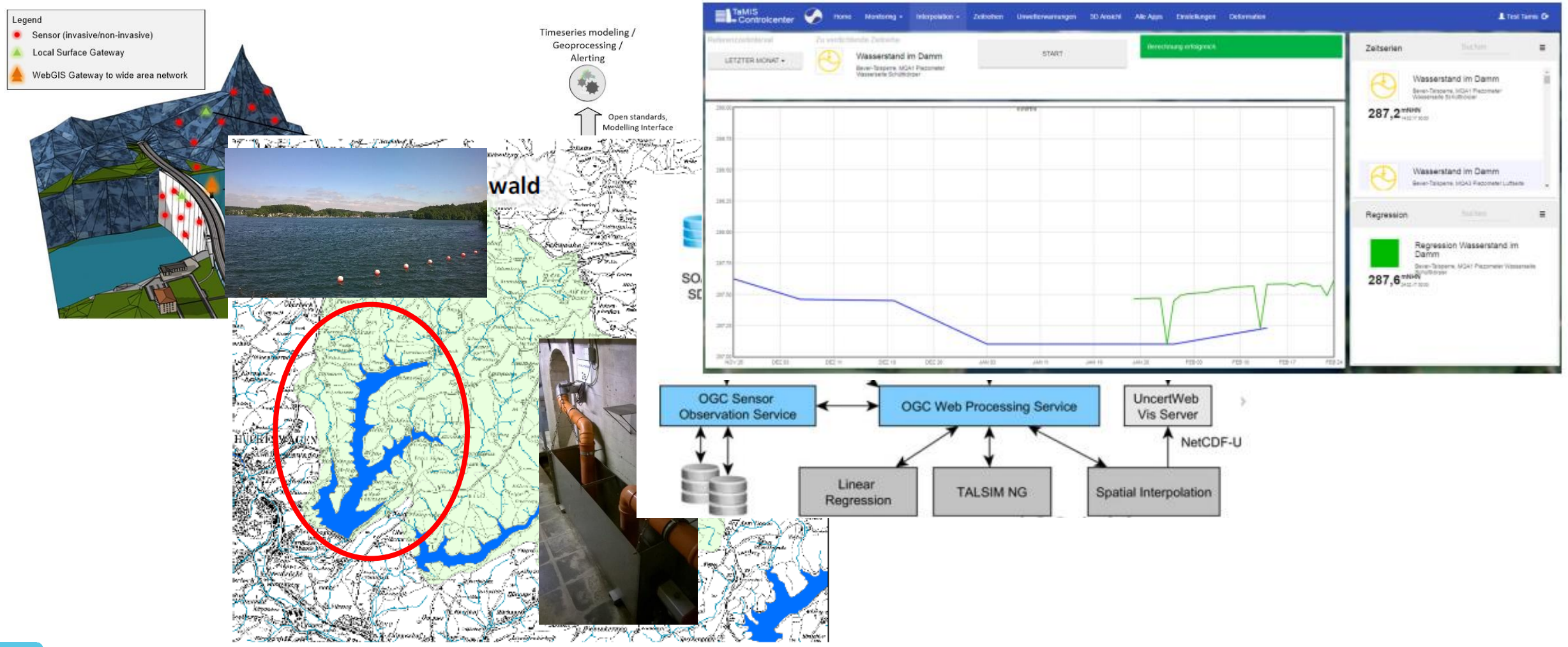
USGS


Sicher | https://cida.usgs.gov/gdp/
Dataset Selection
Search
Algorithms
Data Subsets
Areal Statistics
4km Monthly Parameter-elevation Regressions on Independent Slopes Model Monthly Climate Data for the Continental United States.
Abstract
This dataset was created using the PRISM (Parameter-elevation Regressions on Independent Slopes Mode...
Bias Corrected Constructed Analogs V2 Daily Climate Projections
Abstract
This archive contains projections of daily BCCA CMIP3 and CMIP5 projections of precipitation, daily ...
Bias Corrected Spatially Downscaled Monthly CMIP5 Climate Projections
Abstract
This archive contains 234 projections of monthly BCSD CMIP5 projections of precipitation and monthly...
Bias Corrected Spatially Downscaled Monthly Climate Predictions
Abstract
This archive contains fine spatial-resolution translations of 112 contemporary climate projections o...
CIG Northern US Rockies and Pacific Northwest Statistical Downscaling
Abstract
The goal of this project was to (1) develop consistent historical and future downscaled climate and ...
California Basin Characterization Model Downscaled Climate and Hydrology
Abstract
The California Basin Characterization Model (CA-BCM 2014) dataset provides historical and projected ...
CANADA
Vancouver
Toronto
Chicago
New York
San Francisco
UNITED STATES
Los Angeles
MEXICO
Mexico City
Arctic Ocean
Pacific Ocean
Atlantic Ocean
NORTH AMERICA
EUROPE
AFRICA
SOUTH AMERICA
AUSTRALIA
Washington
Montana
Oregon
Idaho
Wyoming
Denver

EXAMPLE APPLICATION: TAMIS

GEOTECHNOLOGIEN

Bundesministerium für Bildung und Forschung

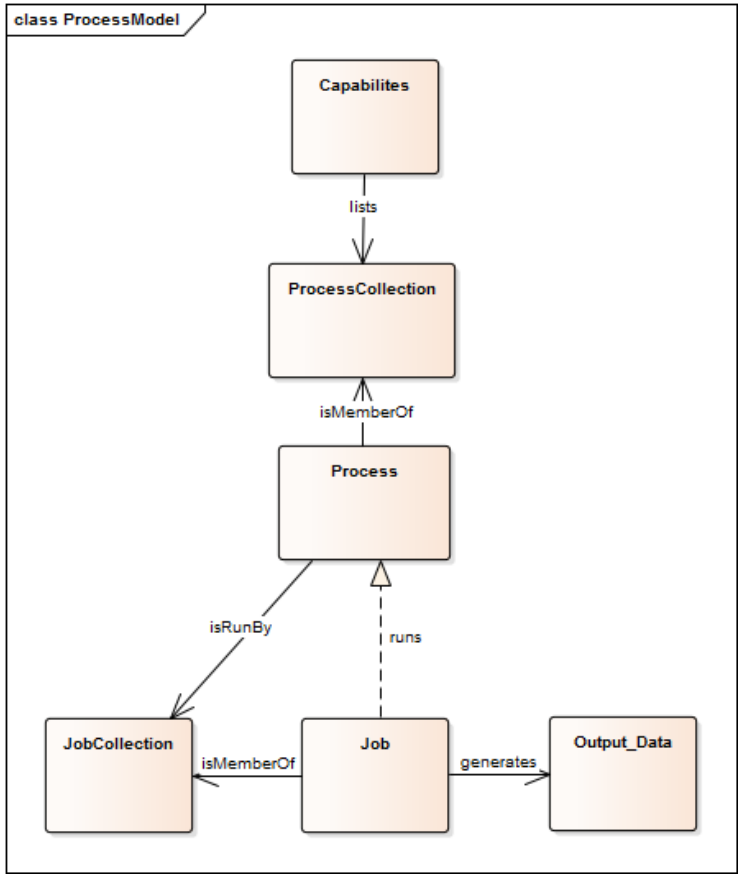


Current WPS topics

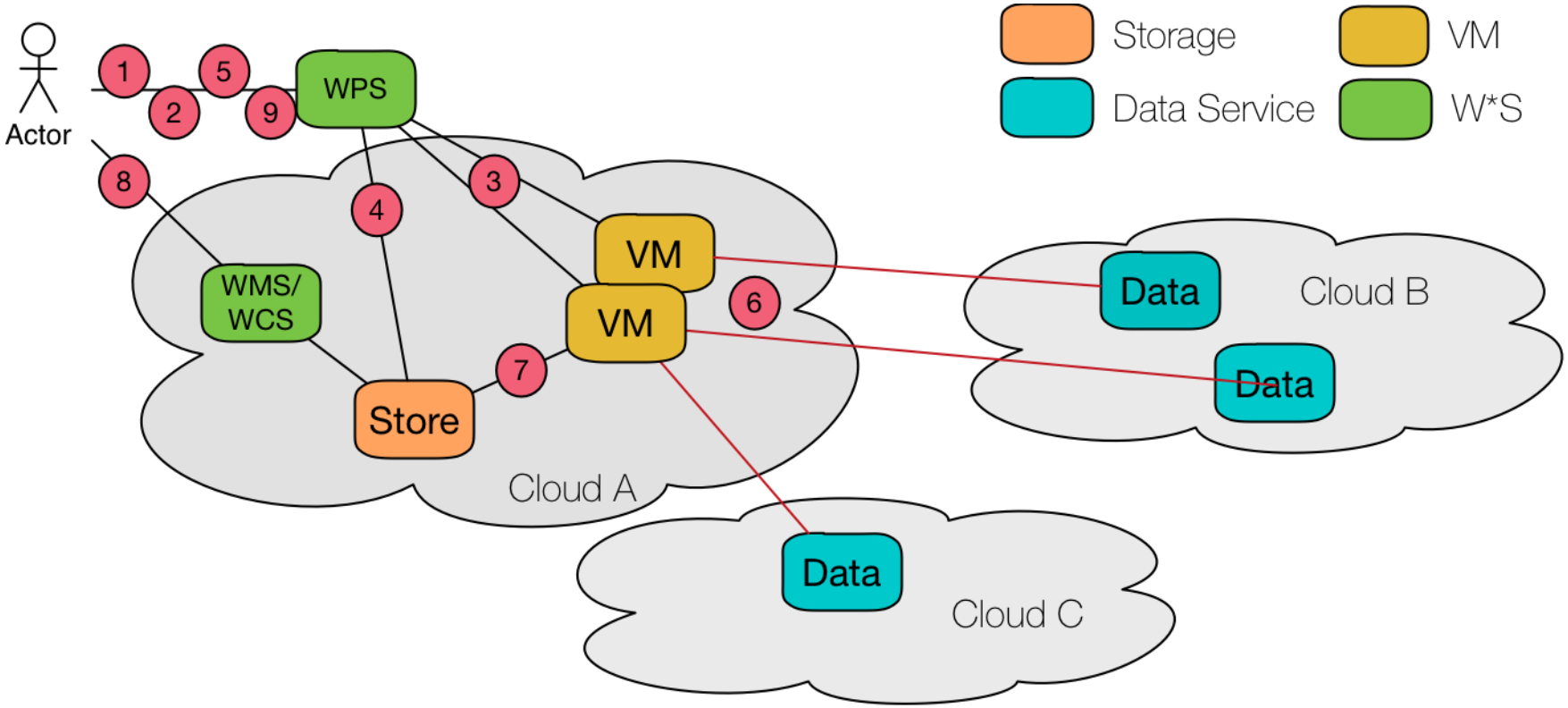
REST API

- 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

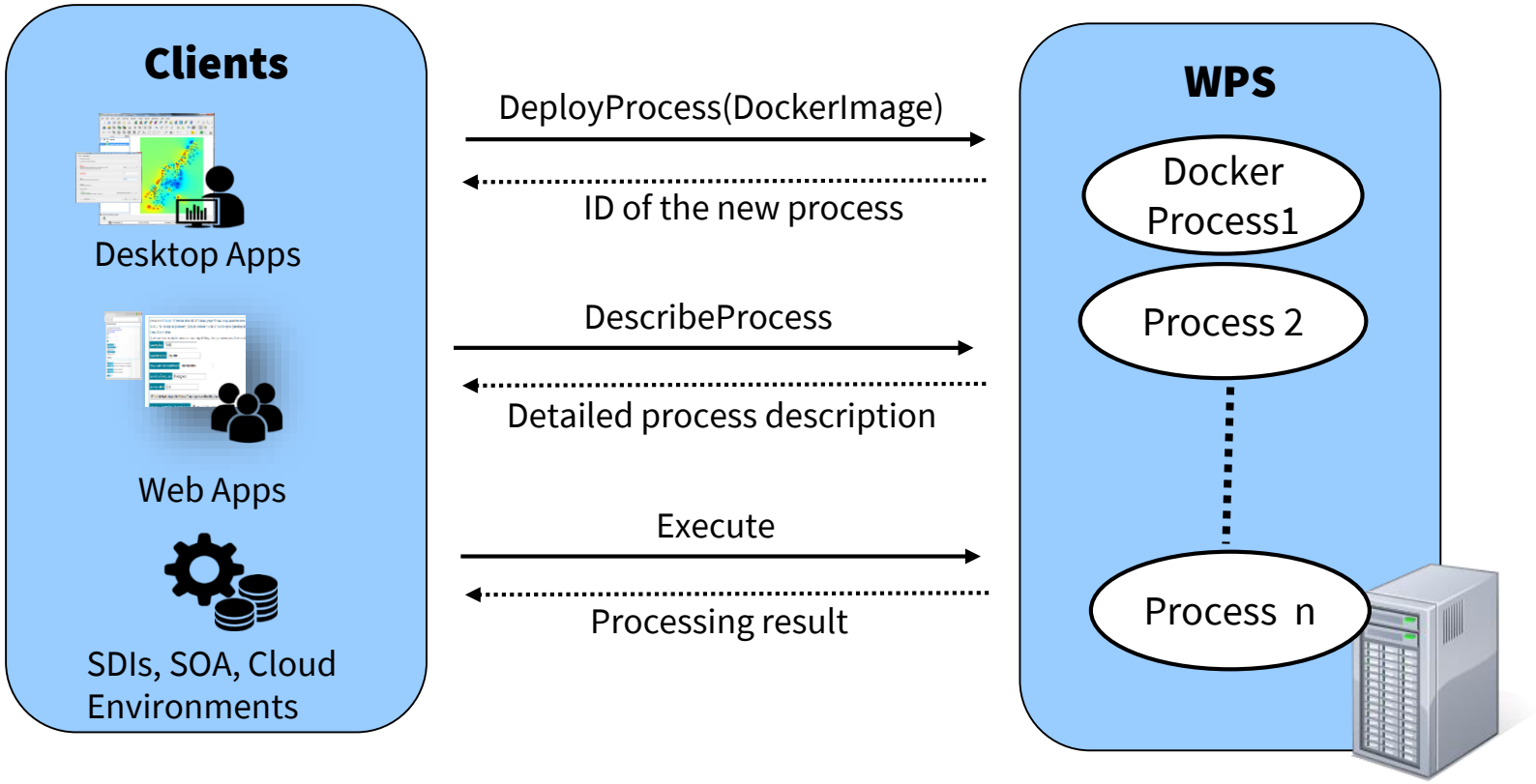


Interface for Cloud-based Geprocessing



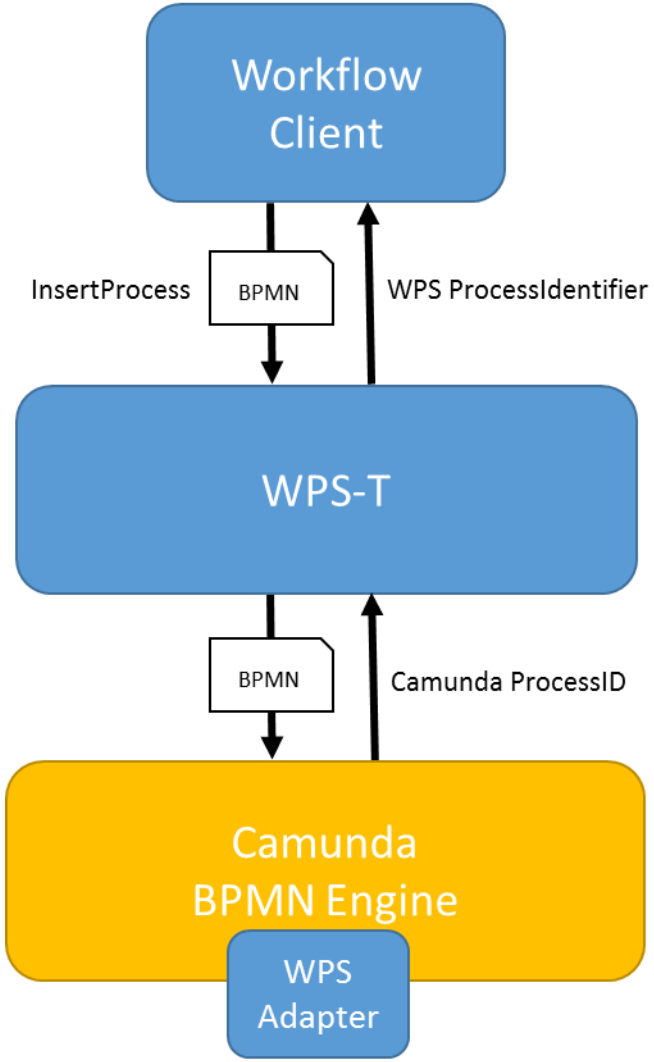
Quelle: http://www.opengeospatial.org/node/2526#Cloud

Hosted Processing



Interoperable Geoprocessing Workflows

- BPMN/Description Languages
- Provenance/Metadata (Discovery)
- Workflow Validation
- Uncertainty Propagation
- Security
- Granularity (Microservices)



52north
exploring horizons

Server Implementations: How can I provide geoprocessing functionality in a WPS?

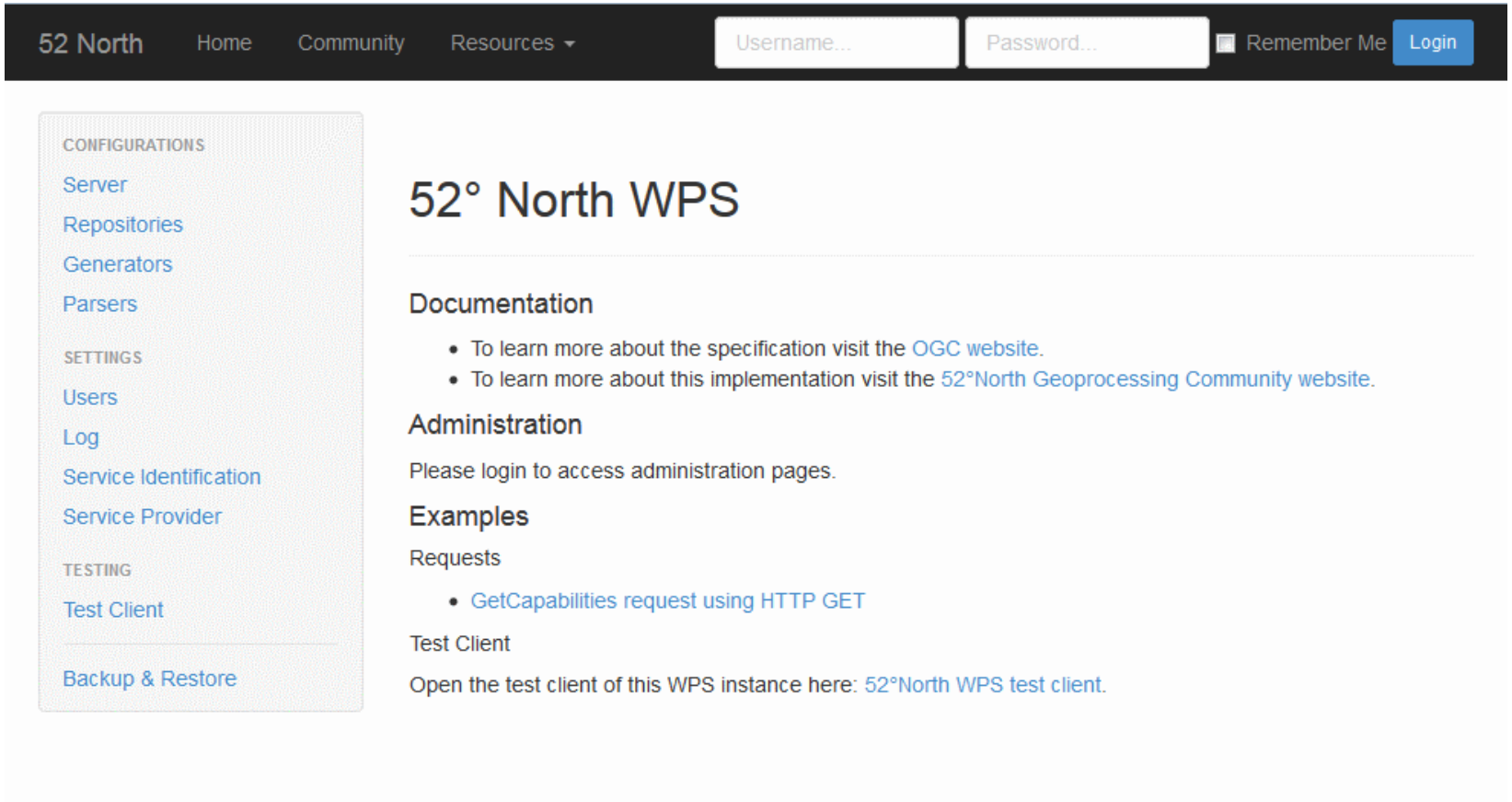
Server Implementations

- OGC lists 54 implementations
- Comercial:
  - 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

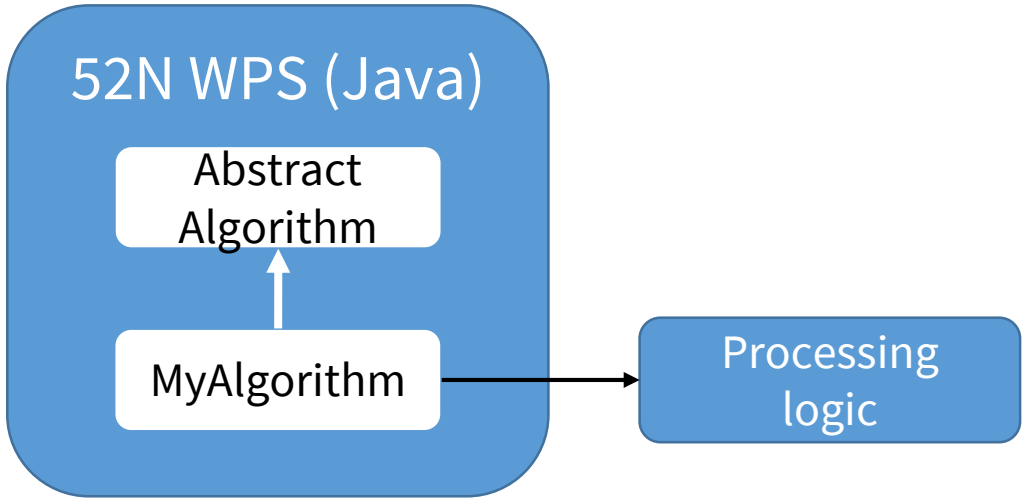


52°North WPS Deployment Patterns

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

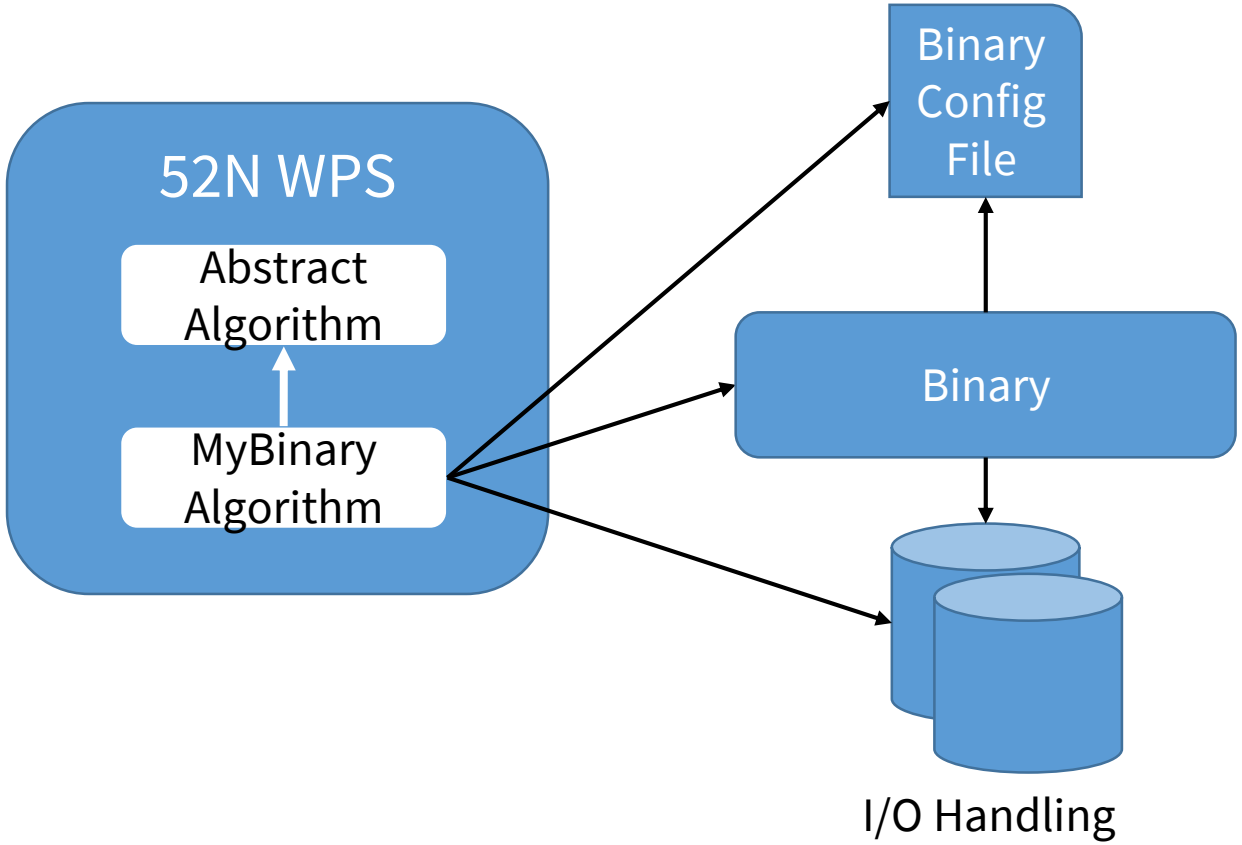
WPS EXTENSION

- 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

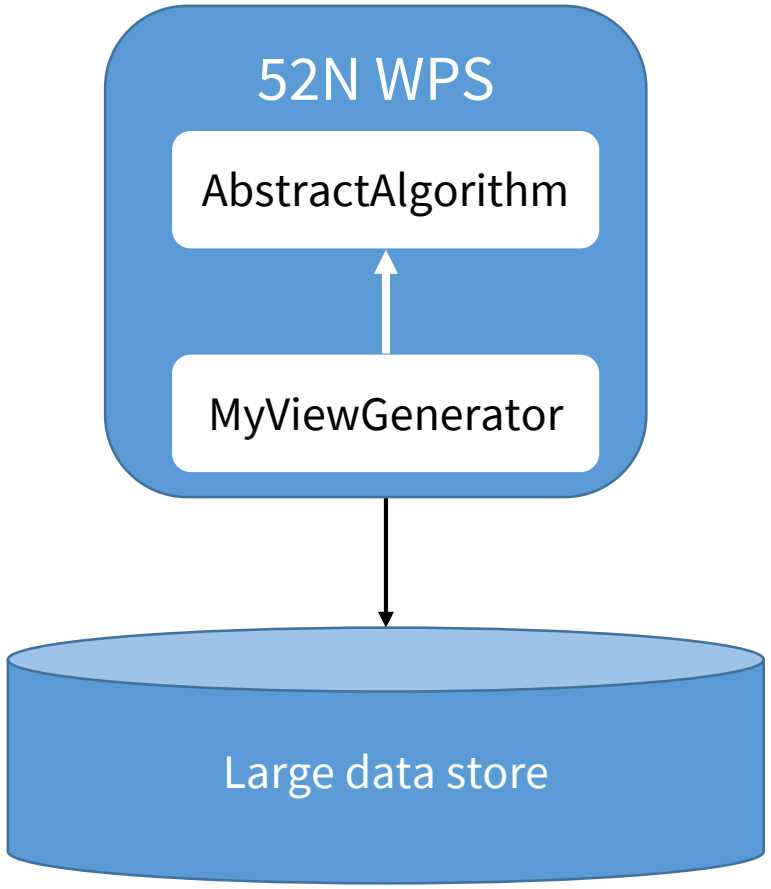
- Execution of a program in binary code
- Configuration via config files
  - Parameters for controlling the process
  - Inputs/Outputs are stored locally and are referenced using config parameters



WPS as Rich Data Interface

- WPS as interface for a large data store
- Dynamic views can be generated using input parameters

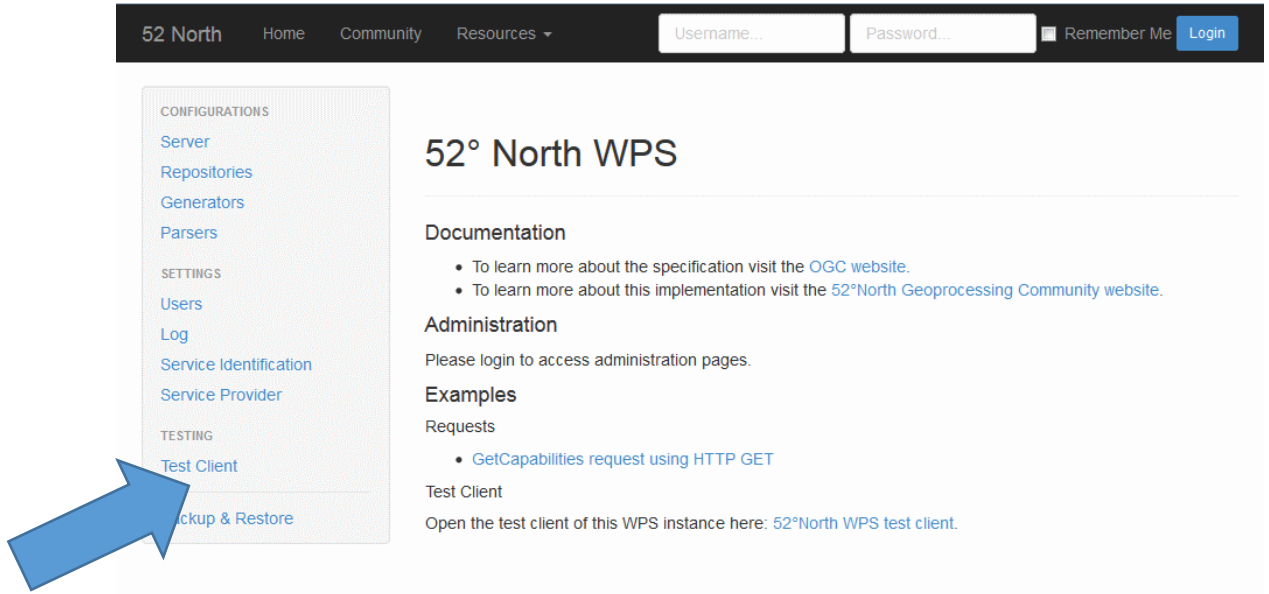
→ Inputs don't contain data for processing



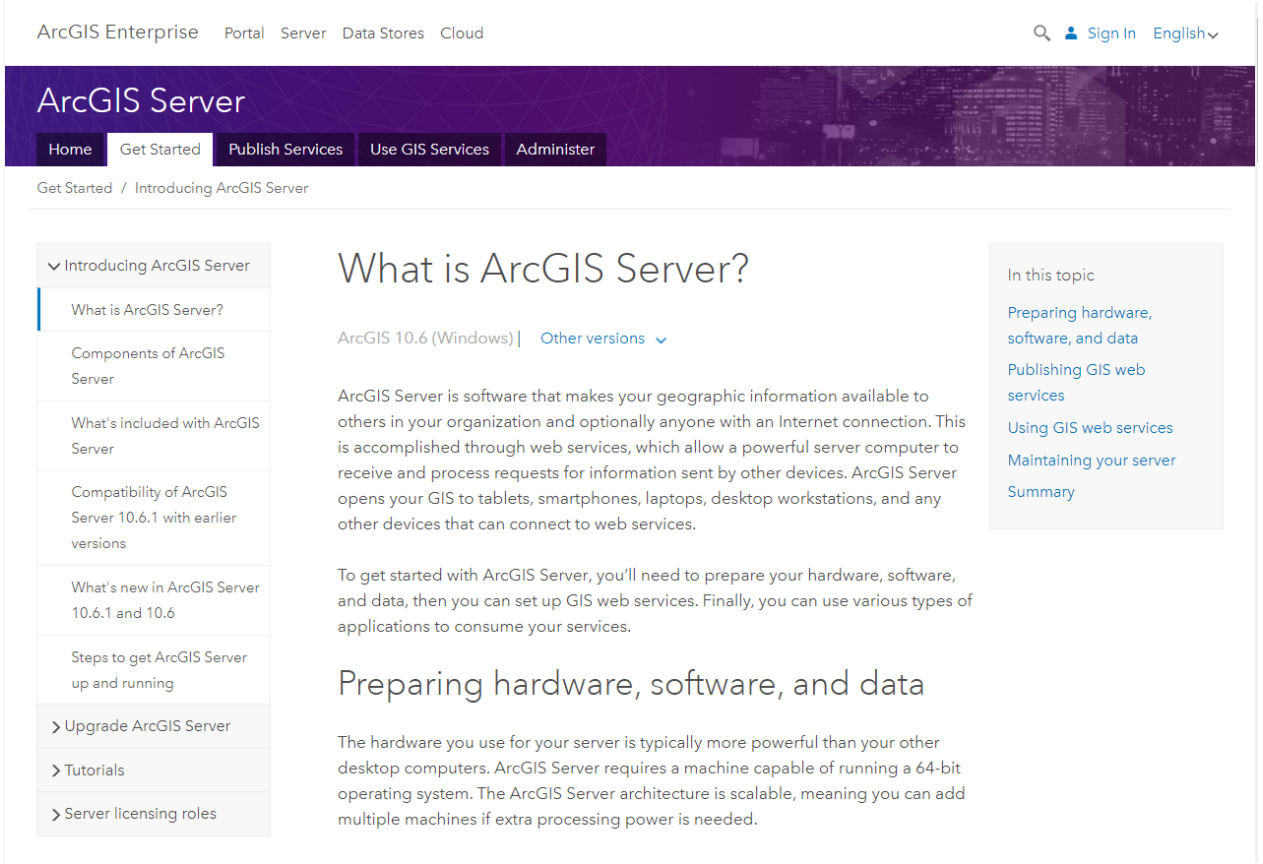
Deployment of R Scripts

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

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



ArcGIS Server WPS Extension



http://enterprise.arcgis.com/de/server/latest/get-started/windows/tutorial-publishing-a-wps-service.htm

52north
exploring horizons

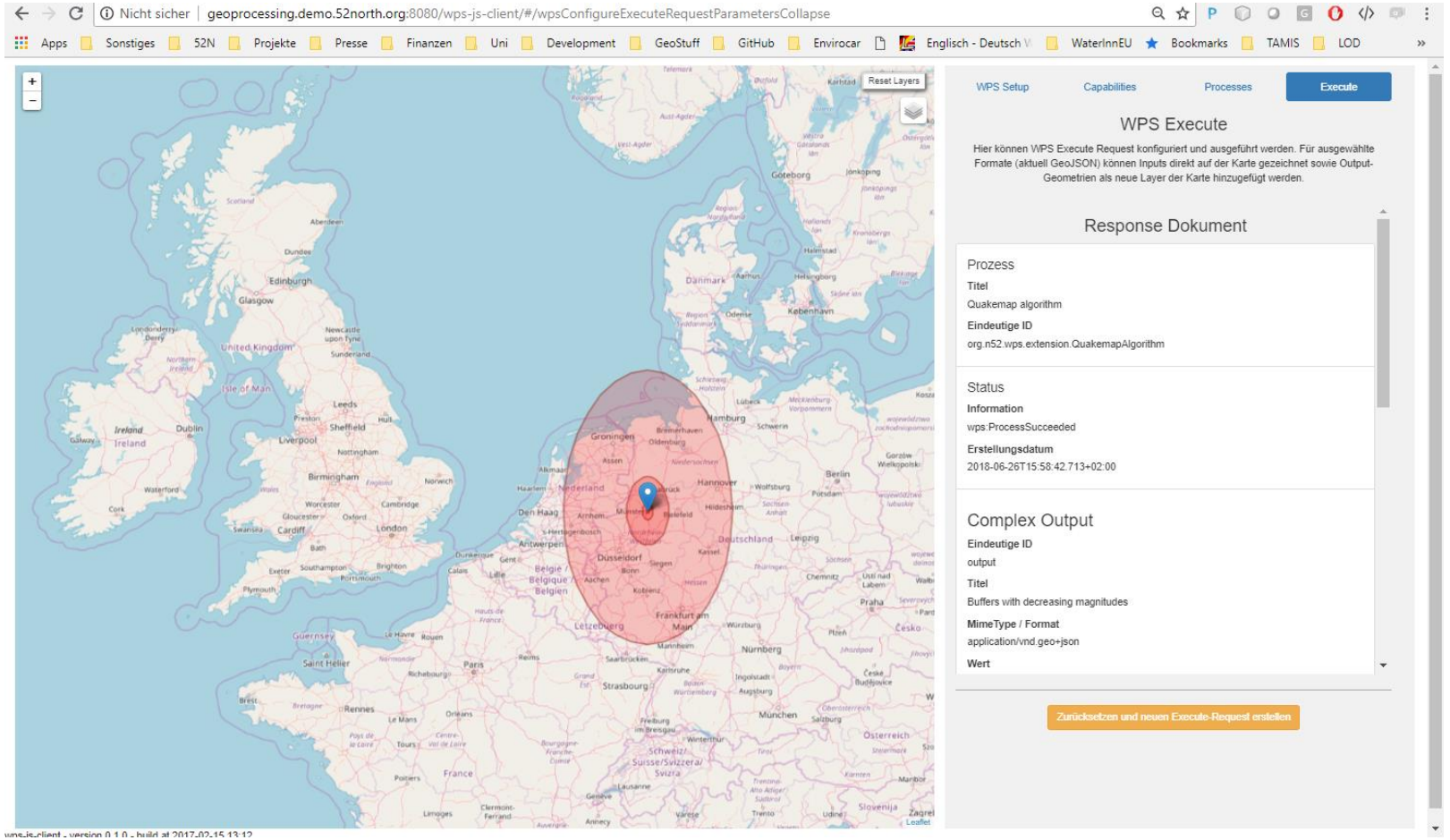
Clients:
How can I use geoprocessing functionality of a WPS?

Client Implementations

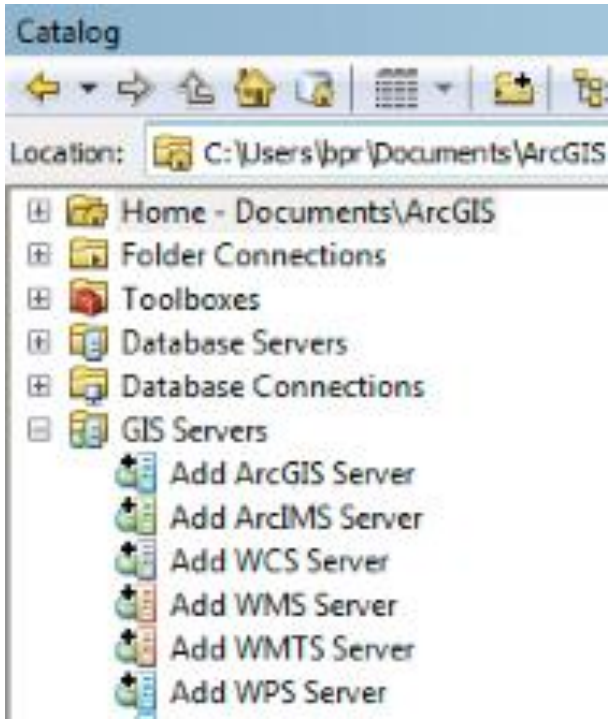
- 52°North ArcGIS WPS Client
  - ArcGIS Extension in collaboration with ESRI Inc.
  - Available as Open Source
- 52°North WPS-JS
  - JavaScript library for the creation of Web clients
- Quantum-GIS WPS – Plugin
- Custom clients for specific applications, e.g.USGS

52°NORTH WPS-JS-CLIENT

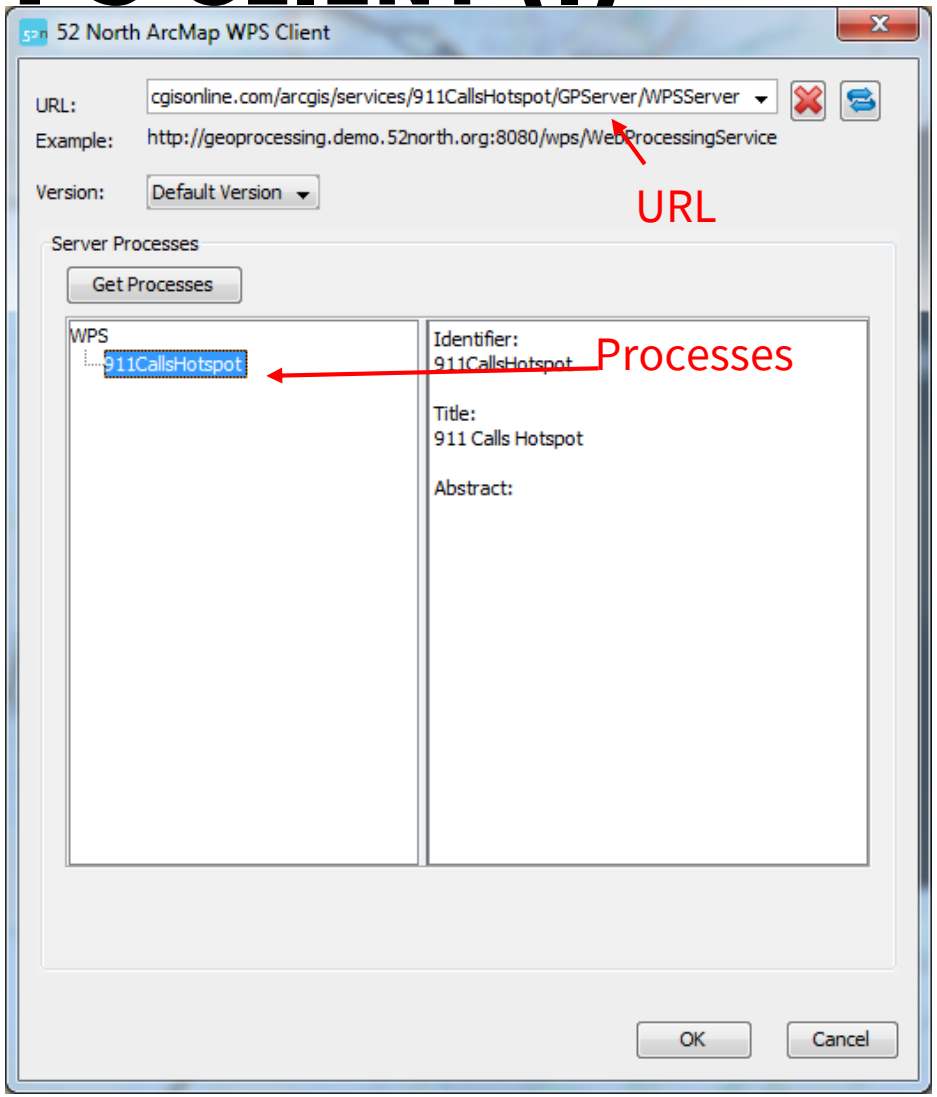
- wps-js:
  - JavaScript client library
- Wps-js-client:
  - JavaScript client
  - Based on the Angular Framework



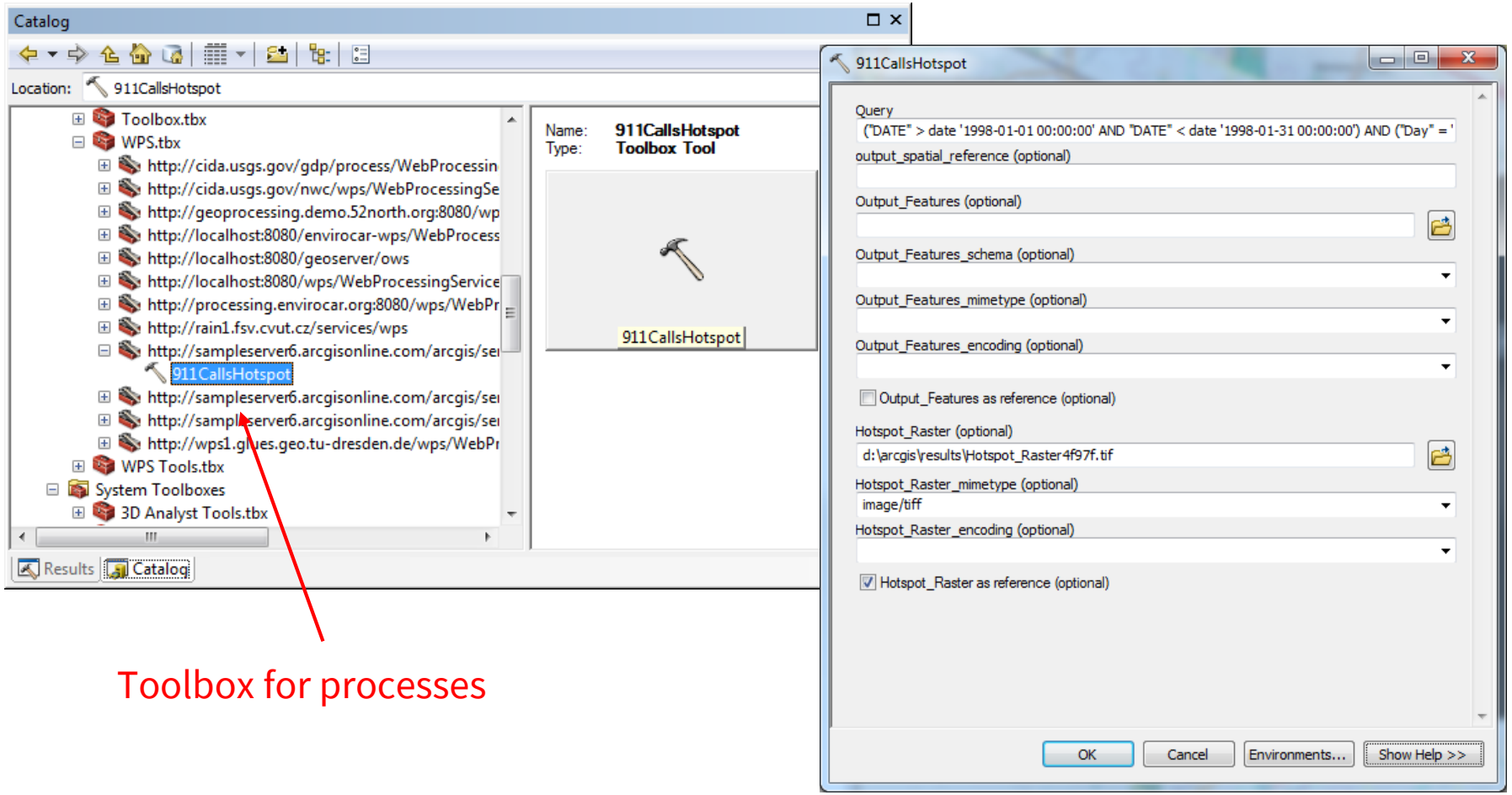
52°North ArcGIS WPS Client (I)



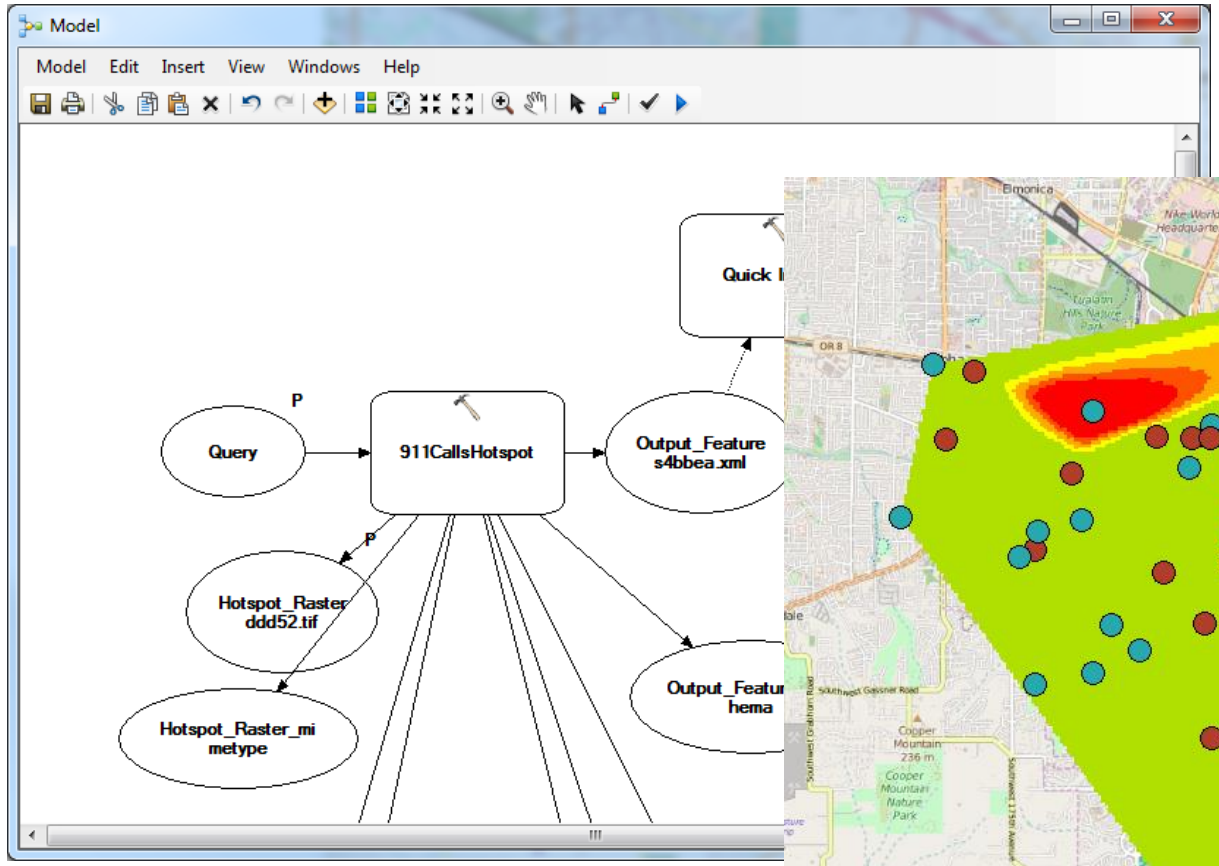
Adding of a WPS



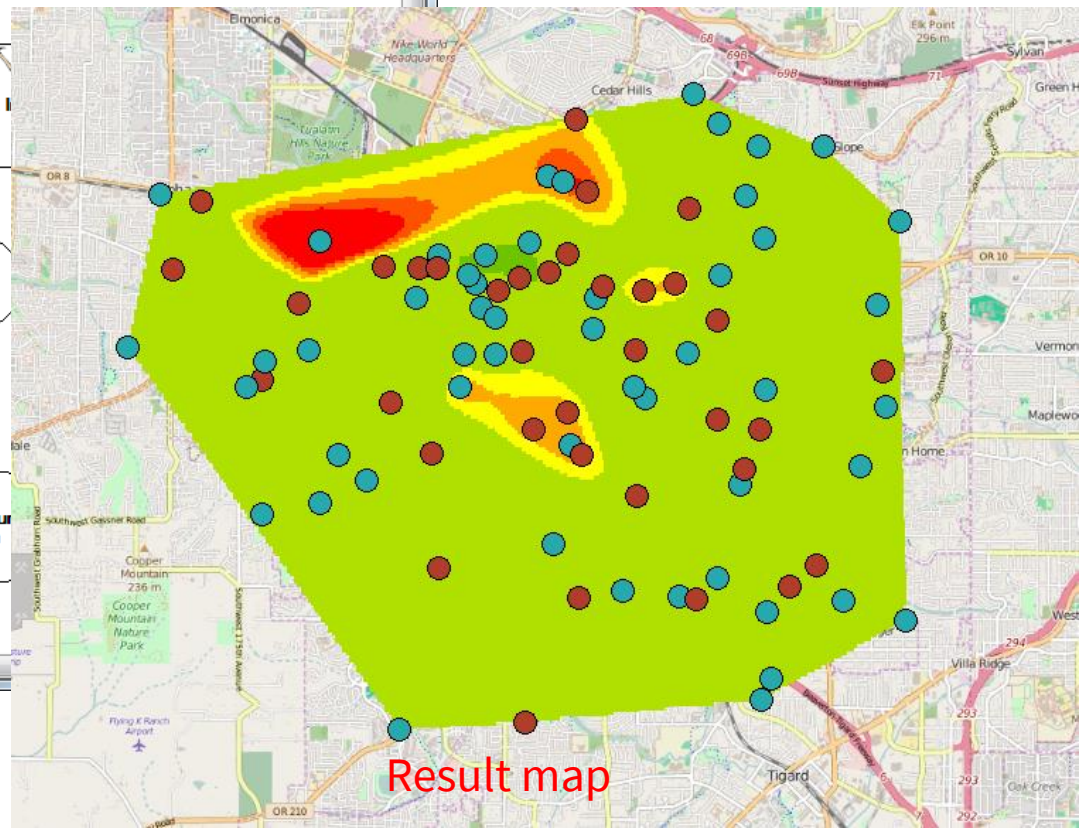
52°NORTH ARCGIS WPS CLIENT (II)



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)

52north
exploring horizons

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/

52N WPS Mailinglist: https://list.52north.org/mailman/listinfo/geoprocessing

Short DEMO

Annex: WPS REST Extension

Motivation For a REST Extension

- WPS 2.0 and otherOGC 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 ressources (not on operations)
  - Usa 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

Testbed-12 REST Architecture Engineering Report

Publication Date: 2017-05-12

Approval Date: 2016-12-07

Posted Date: 2016-10-28

Reference number of this document: OGC 16-035

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A005-1

Category: Public Engineering Report

Editors: Christoph Stasch, Simon Jirka

Title: **Testbed-12 REST Architecture Engineering Report**

OGC Engineering Report

COPYRIGHT

Copyright © 2017 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

WARNING

This document is an OGC Public Engineering Report created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard and not an official position of the OGC membership.It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

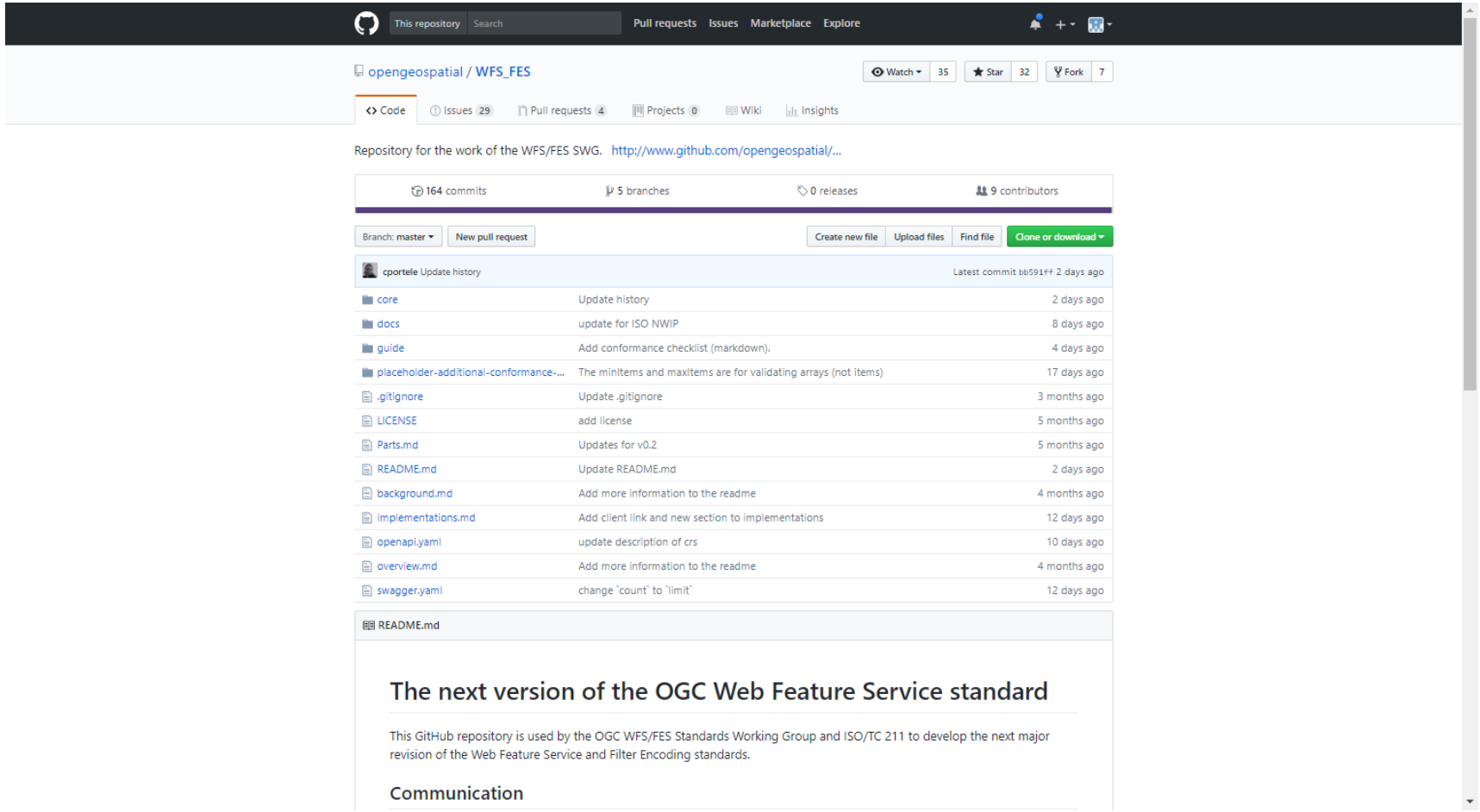
LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modifiedthe Intellectual Property, and to permit persons to whom the Intellectual Property

http://docs.opengeospatial.org/per/16-035.html

EXAMPLE: WEB FEATURE SERVICE 3.0 (DRAFT)



https://github.com/opengeospatial/WFS_FES

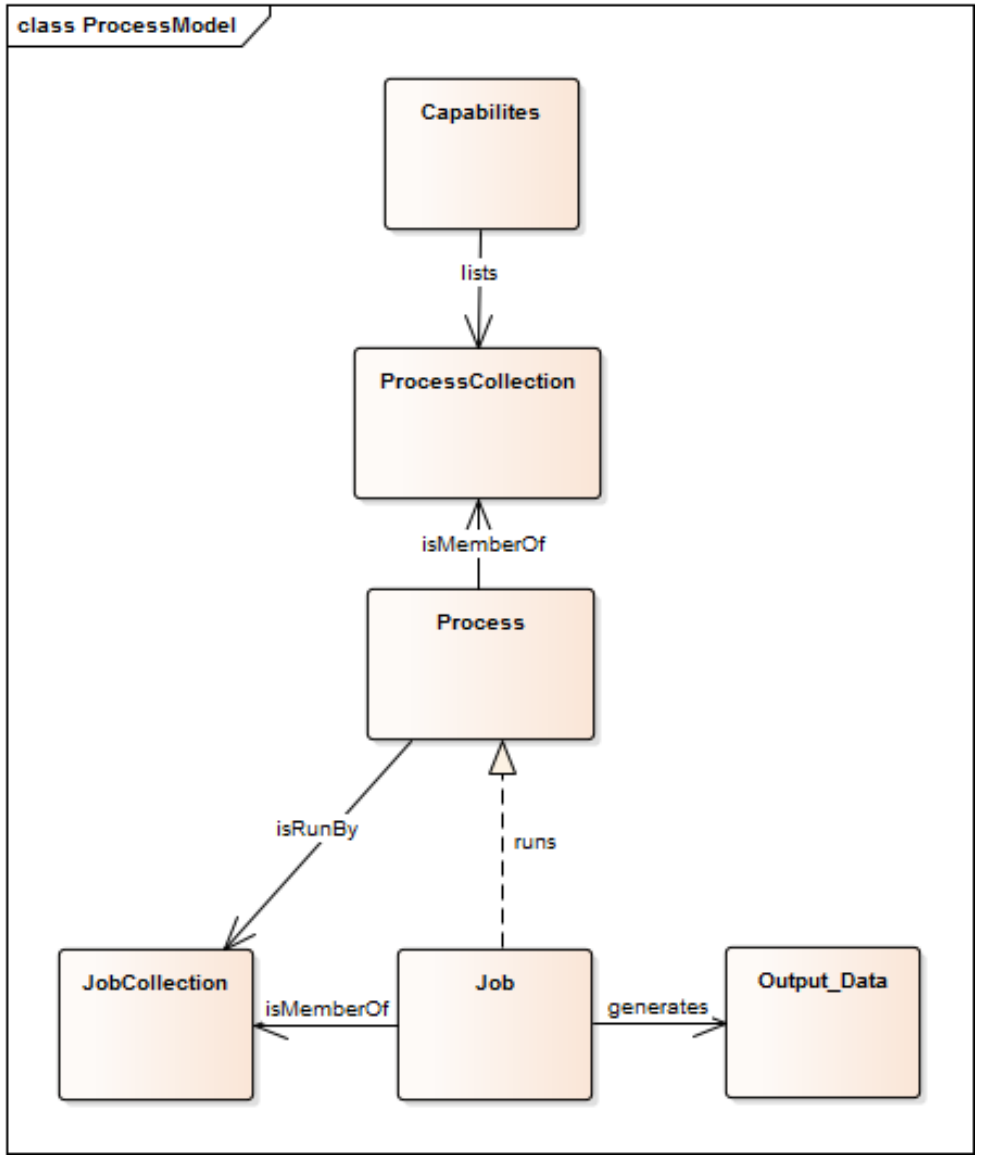
Existing Geoprocessing REST APIs

- 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 Ressource 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
  - …

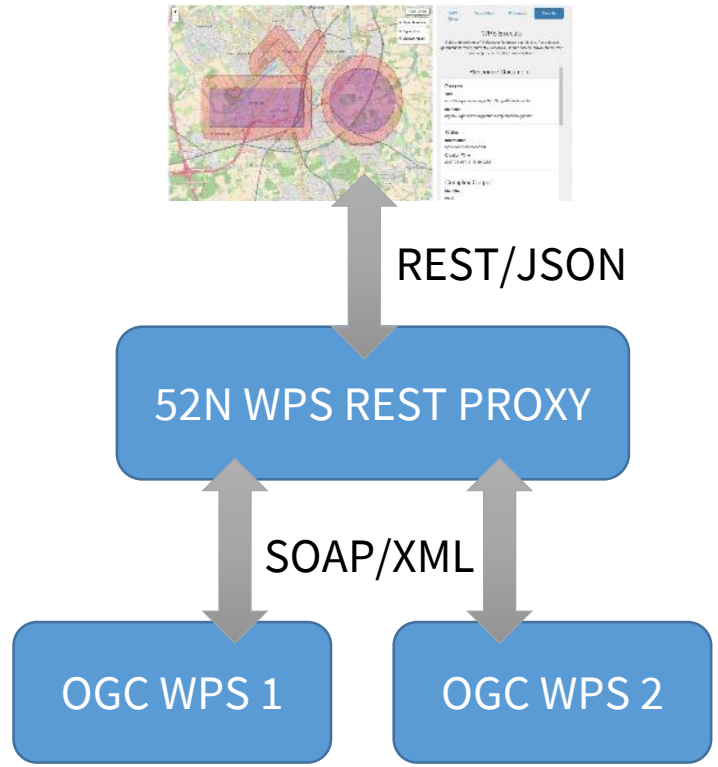


Endpoints for resource access

Ressource	Beschreibung	HTTP Operation	Endpunkt	Message Body
Capabilities	Request of the service description	HTTP GET	{WPSRestBasisURL}	-
ProcessCollection	Request of the list of processes	HTTP GET	{WPSRestBasisURL}/processes	-
Process	Request of a single process description	HTTP GET	{WPSRestBasisURL}/processes/{processID}	-
JobCollection	Request of the list of jobs (executions) of a process	HTTP GET	{WPSRestBasisURL}/processes/{processID}/jobs	-
Job	Execution of a process/creation of a new job	HTTP POST	{WPSRestBasisURL}/processes/{processID}/jobs	Execute Request in JSON
Job	Request of the status of a job	HTTP GET	{WPSRestBasisURL}/processes/{processID}/jobs/{jobID}	-
Outputs	Request of the results of a job	HTTP GET	{WPSRestBasisURL}/processes/{processID}/jobs/{jobID}/results	-

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


```
{
  "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": "52North",
      ...
      }
    },
    "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.algorithm.DouglasPeuckerAlgorithm"
      },...
      ]
    },
    "_service": "WPS",
    "_version": "2.0.0"
  }
}
```


General service information

Process list

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


```
{
  "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.algorithm.JTSConvexHullAlgorithm"
    },...
    ]
}
```


Link to detailed process description

`http://geoprocessing.demo.52north.org:8080/wps-proxy/processes`

REQEST PROCESS DESCRIPTION

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

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


```
{
  "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.geo+json"
              },...
            ]
          },
          "_minOccurs": "1",
          "_maxOccurs": "1"
        }
      ],
      "Output": [
        {
          "Title": "result",
          "Identifier": "result",
          "ComplexData": {
            "Format": [
              {
                "_default": "false",
                "_mimeType": "application/vnd.geo+json"
              },...
            ]
          }
        }
      ]
    },
    "_processVersion": "1.1.0",
    "_jobControlOptions": "sync-execute async-execute",
    "_outputTransmission": "value reference",
    "execute-url": "http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/org.n52.wps.server.algorithm.JTSConvexHullAlgorithm/jobs"
  }
}
```


Input parameter

Output parameter

Execution-URL

`http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/org.n52.wps.server.algorithm.JTSConvexHullAlgorithm`

Execution of a process (I)

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 ressource (process execution) is created

```
{
    "Execute": {
        "Identifier": "org.n52.wps.server.algorithm.JTSConvexHullAlgorithm",
        "Input": [
            {
            "ComplexData": {
                    "_mimeType": "application/wkt",
                    "_text": "POLYGON((847666.55940505 6793166.084248,
                    ...,
                    847666.55940505 6793166.084248))"
            },
            "_id": "data"
            }
        ],
        "output":[{
            "_mimeType": "application/wkt",
            "_id": "result",
            "_transmission": "value"
        }],
        "_service": "WPS",
        "_version": "2.0.0"
    }
}
```

Process-ID

Input data

Desired outputs

`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 ressource
  - After execution is finished link to results

Running job

```
{
  "StatusInfo": {
    "JobID": "c731d14b-1de6-499c-9317-20224e056012",
    "Status": "Running",
    "Progress": 0
  }
}
```

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

```
{
  "StatusInfo": {
    "JobID": "c731d14b-1de6-499c-9317-20224e056012",
    "Status": "Succeeded",
    "Output": "http://geoprocessing.demo.52north.org:8080/wps-proxy/processes/
    org.n52.wps.server.algorithm.JTSConvexHullAlgorithm/jobs/
    c731d14b-1de6-499c-9317-20224e056012/outputs"
  }
}
```

Finished job with link to results

Request processing results

HTTP GET:

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

- Returns JSON result document


```
{
  "Result": {
    "JobID": "2b21bc7a-18d3-4f6e-8046-680c4c69245d",
    "Output": [
      {
        "ID": "result",
        "ComplexData": {
          "_mimeType": "application/wkt",
          "_text": "POLYGON ((849873.67859648 6792439.9324794,
          ...,
          849873.67859648 6792439.9324794))"
        }
      }
    ]
  }
}
```

WPS TESTCLIENT

Testclient verfügbar unter: http://geoprocessing.demo.52north.org:8080/wps/test_client