**Proposal for aN**

**ESGF WPS Extension specification**

**foR a *MultiModelAverage* Service**

Version 3 – March 18, 2015

The Earth System Grid Federation (ESGF) Compute Working Team (CWT) is working to improve interoperability and compute capabilities within the federation using Web services technology. A use case has been targeted wherein averages are computed over data from multiple distributed sources (***MultiModelAverage***). It is hoped that this ***MultiModelAverage*** use case will establish an approach for capturing future compute capabilities. The ESGF-CWT has selected the Open Geographic Consortium (OGC) Web Processing Service (WPS) standard to ensure machine-to-machine interoperability.

This document lays out a proposal for an ***ESGF WPS Extension Specification*** and an accompanying example use of the specification to access a hypothetical WPS-compliant ***MultiModelAverage*** service. The purpose of this document is to generate discussions about its advantages and disadvantages while provoking the design of a federated compute product within the ESGF infrastructure.

**Background**

Figure 1 is an illustration showing a basic Web services architecture. Notable features include the following:

On the server side…

* An ESGF compute node is a system 100 comprising data and/or services 101-106.
* Compute node capabilities are exposed through a Web service interface 107.
* The Web service interface maps compute node capabilities to out-facing HTTP-based communication links that enable external applications to access node capabilities 108.

On the client side…

* Client applications 113 can access server-side capabilities by directly consuming raw Web service endpoints 116. In this case, the Web service interface functions as a server-side API.
* There can also exist client-side APIs 109 that consume raw Web service endpoints 110 and wrap them into functions that are easier to use than raw endpoints 111, 112. This client-side API is a specialized client application. The client-side API (ESGF API) and a client application do not form a client/server relationship. The client application gains access to the ESGF API’s capabilities by binding to the API’s library functions. Client-side API functions can be exposed to a client application through calls 114, 115 to the API's libraries 111, 112.

Therefore…

* There can be two API’s in the system: a server-side API 108 and a client-side API 109.
* A client application can use one, the other, or both 114-116.
* The Web service API can conform to a standard (such as the OGC WPS standard).

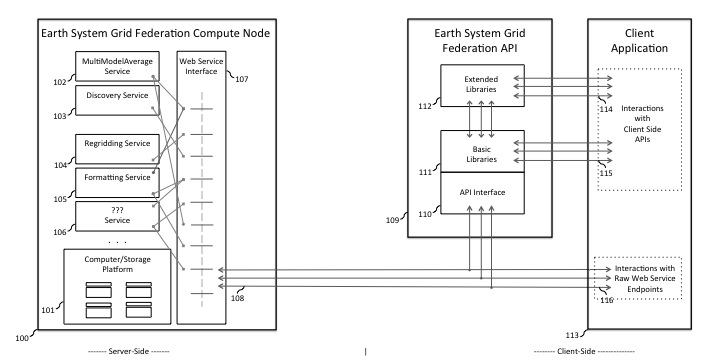


Figure 1. Notional Web services architecture for an ESGF MultiModelAverage service

* The library functions that wrap endpoints in a client-side API generally support in their design, syntax, and semantics the specific needs of a designated community, such as the ESGF. They can include basic functions 111 or more complex programs and workflows 112 that draw on the basic functions.
* Note that the ESGF API is a client, not a server. Client applications, such as UV-CDAT, GrADS, Ferret, IDL, etc. will be able to call on (1) the Web service API or (2) the ESGF API only if the application is (1) compliant with WPS and the ESGF WPS extensions or (2) compliant with the ESGF API.

**Standardization**

Interoperability and accessibility can be improved by defining ESGF community standards at one or more places in a Web services architecture:

1. ESGF could define an **ESGF Compute Node Service Specification** — an agreed upon capability and naming convention for each conformant compute node, e.g., esgf\_Get(params), esgf\_Regrid(params), esgf\_Anomaly(params), esgf\_MultiModelAverage(params), etc. This specification itself would define explicit inputs and outputs. Regardless of how the services are accessed, each node would have known capabilities implemented in known ways.

***AND/OR***

1. ESGF could define an **ESGF WPS Extension Specification** — a specialization of the WPS standard wherein the syntax and semantics of required and optional fields of WPS response documents are tailored to the needs of the ESGF. Here, regardless of how services are implemented or named, their means of access is commonly understood.

***AND/OR***

1. ESGF could define an **ESGF API** — a client-side API that consumes the html Web service endpoints exposed by a WPS-compliant ESGF service and presents them to client applications as a library of easy-to-use function calls tailored to the needs of the ESGF community, e.g. Get(node, params), Regrid(node, params), Anomaly(node, params), MultiModelAverage(nodes, params). Regardless of implementation and communication details, programmers can access node capabilities using a familiar programming library.

**DRAFT ESGF WPS Extension Specification  
for a *MultiModelAverage* Service**

## This draft specification for ESGF-specific extensions to the WPS standard is built around the target use case. The *MultiModelAverage* service described here could exist as a stand-alone service or be one of a suite of services offered by a compute node. This example describes an ESGF-specific use of the WPS standard (Option 2 from above); it is not a specification for how the service itself would be implemented (Option 1) or a specification for a client-side ESGF API (Option 3). As a specification for ESGF extensions to the WPS communication standard, it’s not a specification for the service itself (Option 1). The behaviors are designed into the service.

## Use Case: Multi-Model Averaging:

Averaging over multiple models across federated data.

**Steps for the calculation**:

1. Create the average for each of the individual models.
2. Regrid the models to a common grid.
3. Compute the average across all the regridded data.

## The WPS standard consists of the following three methods:

1. **GetCapabilities**() – returns basic information describing services available at a node.
2. **DescribeProcess**() – returns information about the mandatory, optional, and default parameters needed to invoke a particular service, as well as the format of the data inputs and outputs. **GetCapabilities** and **DescribeProcess** are therefore dynamic discovery aids.
3. **Execute**() - invokes the specified service using the information derived from the DescribeProcess method.

In typical use, **GetCapabilities**() is called first to determine the set of services available at a node — in our example, the sole ***MultiModelAverage*** service. **DescribeProcess**() is then called to retrieve the interaction details for that process. Finally, **Execute**() is called with key-value input parameters according to the details provided by **DescribeProcess**(). **Execute**() provides a response that includes the status of the request and the location of the result.

 The remainder of this section proposes details for applying the ESGF use case to the above three WPS methods. This example is illustrated using XML documents, which are explicitly called out in the OCG specification. Note that certain WPS implementations may support the use of other protocols (i.e., JSON). However, we cannot guarantee interoperability across nodes with variations from the specification.

**1) GetCapabilities():**

The **GetCapabilities** method requests details of a node's service offerings, including service metadata and metadata describing the available services.

* **Input**:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Example Value** | **Description** |
| Service | WPS | Service of interest (versus WMS, etc.) |
| Version | 1.0.0 | API version (required by API to allow version control) |
| Request | GetCapabilities | Specific request type (versus DescribeProcess, Execute) |

* **Output**:

The response document that contains metadata describing the available services. For the ESGF use case, the following outputs are provided in the XML response.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Example Value** | **Description** |
| Identifier | MultiModelAvg | Short-hand reference to the available ESGF process |
| Title | ESGF Use Case: Multi-ModelAverage | Title of the available ESGF process |

There may be a number of capabilities returned which represent the various services for each ESGF compute node site.

**2) DescribeProcess():**

The **DescribeProcess** method requests a detailed description of a particular service available at a node. The "Identifier" parameter specifies the service to describe. Details about multiple services can be requested, separated by commas. At least one service must be specified.

* **Input**:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Example Value** | **Description** |
| Service | WPS | Service of interest (versus WMS, etc.) |
| Version | 1.0.0 | API version (required by API to allow version control) |
| Request | DescribeProcess | Specific request type |
| Identifier | MultiModelAvg | Request metadata for the common ESGF use case |

* **Output**:

The response is an XML document containing metadata about each service identified in the call.

**Reserved WPS parameter names include:**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Example Value** | **Description** |
| Identifier | MultiModelAvg | Short-hand reference to the available ESGF process |
| Title | ESGF Use Case: Multi-ModelAverage | Title of the available ESGF process |
| Abstract | Averaging over multiple models across federated data. | Long-hand description of process |

As reserved WPS parameters, these are required responses for each DescribeProcess call. The proposed ESGF-specific responses follow:

**Proposed ESGF-specific inputs:**

The following table shows ESGF-specific inputs that also could be described in the XML response document for a ***MultiModelAverage*** service's **DescribeProcess** method.

The major classes of inputs are represented as table headings:

* Title Parameter name
* Type Parameter data type (e.g., string, float, int, etc.)
* Abstract Longer description of the process input parameter
* Possible\_values Valid values for the process input parameter
* Optional If ‘Y’, the parameter is not mandatory
* List If ‘Y’, a comma-separated list of values is supported (otherwise 1 item)

These inputs comprise the significant ESGF-specific parameters used in this draft *ESGF WPS Extension Specification* to support a *MultiModelAverage* service, but they could become the core ESGF parameter "standard" for many other services. They could also be used in an *ESGF Node Services Specification* (Option 1 above) and/or an *ESGF API Specification* (Option 3). Note that the primary purpose of this table is provide a strawman proposal to spur further detailed discussion. There should be no expectation that this table is completely accurate in its initial form. All specifics need to be determined by the ESGF CWT (or designate).

**Proposed ESGF-Specific WPS Extensions to Support a MultiModelAverage Service.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Title** | **Type** | **Abstract** | **Possible Values** | **Opt.?** | **List?** |
| ~~Dataset~~ | ~~string~~ | ~~Comma separated list of URIs to NetCDF datasets to subset~~ | ~~Comma separated list of URIs to input NetCDF datasets~~ | ~~N~~ | ~~Y~~ |
| Operation | string | The operation to apply to the variable. | avg, max, min, sum, count, var (varies by site) | N | N |
| Variable | dictionary or list of | The variable(s) that you wish to subset.  Dictionary describes URI and variable name | {“uri”:<http://bla>,id=”clt”,”domain”:”last35yrs”} | N | ? |
| Variable.uri | ? | ? | ? | ? | ? |
| Variable.domain | ? | ? | ? | ? | ? |
| Domain | Dictionary  Or list of | Dictionary to describe the domain you wish to extract, keys should be the dimensions names, values should be dictionaries containing “start”,”end”,”system”.  Dimensions not specified are assumed to be retrieved in full. “system” can be one of “values” . An “id” should be given so that variables can refer to these easily w/o too many duplications | {‘id’:’last35yrs’, 'level': {'end': 13, 'start': 0, 'system': 'indices'},  'time': {'end': '2015-03-19', 'start': '1980', 'system': 'values'}} | Y | ? |
| Domain.id | ? | ? | ? | ? | ? |
| Domain.level..start | ? | ? | ? | ? | ? |
| Domain.level.end | ? | ? | ? | ? | ? |
| Domain.level.system | ? | ? | ? | ? | ? |
| Domain.time | ? | ? | ? | ? | ? |
| Domain.time.start | ? | ? | ? | ? | ? |
| Domain.time.end | ? | ? | ? | ? | ? |
| Domain.time.system | ? | ? | ? | ? | ? |
| GridderTool | string | The gridder to use | ESMF, SCRIP, etc. | Y | N |
| GridderMethod | string | The method to use | “linear”, “conserve”,etc… |  |  |
| Grid | string | The desired grid resolution | T85 | Y | N |
| Output | string | Output format | OpendapURL, URI, NetCDF | Y | N |
| OutputFileName | string | Requested name of output file | Any string (e.g., multi-model-avg.nc) | Y | N |

**3) Execute():**

The **Execute** method is a request to a service to perform its operations on specified input values. The request may be made as either a GET URL, or a POST with an XML (or JSON) request document. Because the request has a complex structure, the POST form is more typically used. For the ESGF use case, the output of the **GetCapabilities** request above will serve as the input for the **Execute** operation.

* **Input**:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Example Value** | **Description** |
| Service | WPS | Service of interest (versus WMS, etc.) |
| Version | 1.0.0 | API version (required by API to allow version control) |
| Request | Execute | Specific request type |
| Identifier | MultiModelAvg | Requests execution of the common ESGF use case |
| ResponseForm | ResponseDocument | Requests output in an XML document |
| StoreExecuteResponse | True | Requests that the Execute response is stored as a web-accessible resource |
| Status | True | Requests that status information be updated in the response document |
| DataInputs | ***See DataInputs Table below ...*** | Process-specific parameter values |

***DataInputs Table***

|  |  |
| --- | --- |
| **Title** | **Use Case Value** |
| ~~Dataset~~ | ~~http://opendap.esgf/tasavg\_mm\_ECMWF\_200001-200012.nc,~~  ~~http://opendap.esgf/tasavg\_mm\_CFSR\_200001-200012.nc,~~  ~~http://opendap.esgf/tasavg\_mm\_MERRA\_200001-200012.nc~~ |
| Operation | avg |
| Variable | {"domain": "year\_2001", "id": "tas", "uri": "http://opendap.esgf/tasavg\_mm\_ECMWF\_200001-200012.nc"}, {"domain": "year\_2001", "id": "tas", "uri": "http://opendap.esgf/tasavg\_mm\_CFSR\_200001-200012.nc"}, {"domain": "year\_2001", "id": "tas", "uri": "http://opendap.esgf/tasavg\_mm\_MERRA\_200001-200012.nc"} |
| ~~Start\_date~~ | ~~20000101~~ |
| ~~End\_date~~ | ~~20001231~~ |
| ~~Max\_lat~~ | ~~50~~ |
| ~~Max\_lon~~ | ~~-66~~ |
| ~~Min\_lat~~ | ~~24~~ |
| ~~Min\_lon~~ | ~~-125~~ |
| ~~Start\_level~~ | ~~1~~ |
| ~~End\_level~~ | ~~42~~ |
| Domain | '{"level": {"start": 1, "end": 42, "system": "indices"}, "id": "year\_2001", "time": {"start": "2001-01-01", "end": "2001-12-31", "system": "values"}}' |
| GridderTool | ‘esmf’ |
| GridderMethod | ‘conserve’ |
| Grid | T85 |
| Output | OPeNDAP URL |
| OutputFileName | multi-model-avg-north-america-2000-ECMWF-CFSR-MERRA.nc |

* **Output**:

For the ESGF use case, a NetCDF file will ultimately be returned in an OPeNDAP URL link.

The **Execute** response is a URL that points to an XML (or JSON) document that is updated with status information about the requested process. During process execution, the percentage completion estimate is provided here. When the process is complete, a URL pointing to the process result is added to the document. For the ESGF use case, the following outputs are provided in the XML (or JSON) response.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Example Value** | **Description** |
| Identifier | MultiModelAvg | Short-hand reference to the available ESGF process |
| Title | ESGF Use Case: Multi-ModelAverage | Title of the available ESGF process |
| Abstract | Averaging over multiple models across federated data. | Long-hand description of process |
| StatusLocation | <http://opendap.esgf/WPS?OUTPUT>=  /pywps-142418624003.xml | Web-accessible location of status document |
| Status | ProcessAccepted, ProcessStarted, ProcessSucceeded, ProcessFailed | Overall status of process execution |
| Output Message | Basically stdout and stderr | Verbose output of server side, should contain traceback in case of error. |
| PercentCompleted | 1 (ProcessStarted),  100 (ProcessSucceeded) | Indicates amount of processing completed |
| Output | {“uri”:’https://opendap.esgf/multi-model-avg-north-america-2000-ECMWF-CFSR-MERRA.nc’,’id’:’tas’} | Web-accessible location of process result |

**EXAMPLE USE OF THE DRAFT ESGF WPS Extension Specification for a *MultiModelAverage* Service**

This section converts the English version of the ESGF use case to WPS Syntax. More specifically, to web service URLs. Accordingly, the inputs and outputs are samples based on the parameters defined in the previous section.

**1) GetCapabilities():**

The **GetCapabilities** method requests details of the service offering, including service metadata and metadata describing the available operations. The response is an XML document called the **capabilities document**.

* **Input URL**:

<http://localhost/cgi-bin/pywps.cgi?&>

version=1.0.0&

service=WPS&

request=GetCapabilities

* **Sample Response Document**:

<wps:Capabilities service=*"WPS"* version=*"1.0.0"*

xml:lang=*"en-CA"*

xsi:schemaLocation=*"http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsGetCapabilities\_response.xsd"*

updateSequence=*"1"*>

<ows:ServiceIdentification>

<ows:Title>NASA Center for Climate Simulation - WPS Server</ows:Title>

<ows:Abstract>See https://cds.nccs.nasa.gov</ows:Abstract>

<ows:Keywords>

<ows:Keyword>WPS</ows:Keyword>

</ows:Keywords>

<ows:ServiceType>WPS</ows:ServiceType>

<ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>

<ows:Fees>None</ows:Fees>

<ows:AccessConstraints>none</ows:AccessConstraints>

</ows:ServiceIdentification>

<ows:ServiceProvider>

<ows:ProviderName>NASA GSFC</ows:ProviderName>

<ows:ProviderSite xlink:href=*"http://www.nasa.gov/"* />

<ows:ServiceContact>

<ows:IndividualName>John Schnase, Dan Duffy</ows:IndividualName>

<ows:PositionName>Principal Investigator</ows:PositionName>

<ows:ContactInfo>

<ows:Address>

<ows:DeliveryPoint>8800 Greenbelt Road</ows:DeliveryPoint>

<ows:City>Greenbelt</ows:City>

<ows:PostalCode>20771</ows:PostalCode>

<ows:Country>usa</ows:Country>

<ows:ElectronicMailAddress>login@server.org</ows:ElectronicMailAddress>

</ows:Address>

<ows:OnlineResource xlink:href=*"http://www.nasa.gov/"* />

<ows:HoursOfService>0:00-24:00</ows:HoursOfService>

<ows:ContactInstructions>none</ows:ContactInstructions>

</ows:ContactInfo>

<ows:Role>Technical Lead</ows:Role>

</ows:ServiceContact>

</ows:ServiceProvider>

<ows:OperationsMetadata>

<ows:Operation name=*"GetCapabilities"*>

<ows:DCP>

<ows:HTTP>

<ows:Get xlink:href=*"http://nccs.gov/cgi-bin/wps?"* />

<ows:Post xlink:href=*"http://nccs.gov/cgi-bin/wps"* />

</ows:HTTP>

</ows:DCP>

</ows:Operation>

<ows:Operation name=*"DescribeProcess"*>

<ows:DCP>

<ows:HTTP>

<ows:Get xlink:href=*"http://nccs.gov/cgi-bin/wps?"* />

<ows:Post xlink:href=*"http://nccs.gov/cgi-bin/wps"* />

</ows:HTTP>

</ows:DCP>

</ows:Operation>

<ows:Operation name=*"Execute"*>

<ows:DCP>

<ows:HTTP>

<ows:Get xlink:href=*"http://nccs.gov/cgi-bin/wps?"* />

<ows:Post xlink:href=*"http://nccs.gov/cgi-bin/wps"* />

</ows:HTTP>

</ows:DCP>

</ows:Operation>

</ows:OperationsMetadata>

<wps:ProcessOfferings>

<wps:Process wps:processVersion=*"1.0.0"*>

<ows:Identifier>**MultiModelAvg**</ows:Identifier>

<ows:Title>**ESGF Use Case: Multi-Model Average**</ows:Title>

<ows:Abstract>**Averaging over multiple models across federated data.**</ows:Abstract>

</wps:Process>

</wps:ProcessOfferings>

<wps:Languages>

<wps:Default>

<ows:Language>en-CA</ows:Language>

</wps:Default>

<wps:Supported>

<ows:Language>en-CA</ows:Language>

</wps:Supported>

</wps:Languages>

<wps:WSDL xlink:href=*"http://nccs.gov/cgi-bin/wps?WSDL"* />

</wps:Capabilities>

**2) DescribeProcess():**

The **DescribeProcess** method requests full details about specific services that are available. The parameter **Identifier** specifies the services to be described. Multiple services can be requested, separated by commas. At least one service must be specified.

* **Input URL**:

<http://localhost/cgi-bin/pywps.cgi?&>

version=1.0.0&

service=WPS&

request=DescribeProcess&

identifier=MultiModelAvg

* **Sample Response Document**:

<wps:ProcessDescriptions

xsi:schemaLocation=*"http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsDescribeProcess\_response.xsd"*

service=*"WPS"* version=*"1.0.0"* xml:lang=*"en-CA"*>

<ProcessDescription wps:processVersion=*"1.0.0"*

storeSupported=*"false"* statusSupported=*"false"*>

<ows:Identifier>**MultiModelAvg**</ows:Identifier>

<ows:Title>ESGF Use Case: Multi-Model Average</ows:Title>

<ows:Abstract>Averaging over multiple models across federated data.</ows:Abstract>

<DataInputs>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**End\_level**</ows:Identifier>

<ows:Title>The last vertical level to extract data (degrees less

than or equal to 42)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#integer"*>integer</ows:DataType>

<ows:AnyValue />

<DefaultValue>42</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"1"* maxOccurs=*"1"*>

<ows:Identifier>**End\_date**</ows:Identifier>

<ows:Title>The last date/time to extract data (yyyyMMddHHmmss)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AnyValue />

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Min\_lat**</ows:Identifier>

<ows:Title>Minimum latitude (degrees greater than or equal to -90.0)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#float"*>float</ows:DataType>

<ows:AnyValue />

<DefaultValue>-90.0</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"1"* maxOccurs=*"1"*>

<ows:Identifier>**OutputFileName**</ows:Identifier>

<ows:Title>Requested name of output file</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AnyValue />

</LiteralData>

</Input>

<Input minOccurs=*"2"* maxOccurs=*"10"*>

<ows:Identifier>**Dataset**</ows:Identifier>

<ows:Title>Comma separated list of URIs to NetCDF datasets to subset</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AnyValue />

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Start\_level**</ows:Identifier>

<ows:Title>The first vertical level to extract data (degrees greater

than or equal to 1)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#integer"*>integer</ows:DataType>

<ows:AnyValue />

<DefaultValue>1</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Output**</ows:Identifier>

<ows:Title>Output format</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AllowedValues>

<ows:Value>OpendapURL</ows:Value>

<ows:Value>URI</ows:Value>

</ows:AllowedValues>

<DefaultValue>OpendapURL</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Grid**</ows:Identifier>

<ows:Title>The desired grid resolution</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AllowedValues>

<ows:Value>T85</ows:Value>

</ows:AllowedValues>

<DefaultValue>T85</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Max\_lat**</ows:Identifier>

<ows:Title>Maximum latitude (degrees less than or equal to 90.0)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#float"*>float</ows:DataType>

<ows:AnyValue />

<DefaultValue>90.0</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"1"* maxOccurs=*"1"*>

<ows:Identifier>**Variable**</ows:Identifier>

<ows:Title>The variable that you wish to subset..</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AllowedValues>

<ows:Value>tas</ows:Value>

<ows:Value>pr</ows:Value>

<ows:Value>ta</ows:Value>

</ows:AllowedValues>

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Min\_lon**</ows:Identifier>

<ows:Title>Minimum longitude (degrees greater than or equal to

-180.0)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#float"*>float</ows:DataType>

<ows:AnyValue />

<DefaultValue>-180.0</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Operation**</ows:Identifier>

<ows:Title>The operation to apply to the variable.</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AllowedValues>

<ows:Value>avg</ows:Value>

<ows:Value>max</ows:Value>

<ows:Value>min</ows:Value>

<ows:Value>sum</ows:Value>

<ows:Value>count</ows:Value>

<ows:Value>var</ows:Value>

</ows:AllowedValues>

<DefaultValue>avg</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Max\_lon**</ows:Identifier>

<ows:Title>Maximum longitude (degrees less than or equal to 180.0)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#float"*>float</ows:DataType>

<ows:AnyValue />

<DefaultValue>180.0</DefaultValue>

</LiteralData>

</Input>

<Input minOccurs=*"1"* maxOccurs=*"1"*>

<ows:Identifier>**Start\_date**</ows:Identifier>

<ows:Title>The first date/time to extract data (yyyyMMddHHmmss)</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AnyValue />

</LiteralData>

</Input>

<Input minOccurs=*"0"* maxOccurs=*"1"*>

<ows:Identifier>**Gridder**</ows:Identifier>

<ows:Title>The gridder to use</ows:Title>

<LiteralData>

<ows:DataType ows:reference=*"http://www.w3.org/TR/xmlschema-2/#string"*>string</ows:DataType>

<ows:AllowedValues>

<ows:Value>EMSF</ows:Value>

</ows:AllowedValues>

<DefaultValue>EMSF</DefaultValue>

</LiteralData>

</Input>

</DataInputs>

</ProcessDescription>

</wps:ProcessDescriptions>

**3) Execute():**

The **Execute** method is a request to invoke the service and perform its operations with specified input values and required output data items. The request may be made as either a GET URL, or a POST with an XMLrequest document. When the request has a complex structure, the POST form is more typically used; however, both request types must be supported according to the specification

The inputs and outputs required for the request depend on the service being executed. For the ESGF use case, the output of the **GetCapabilities** request above will serve as the input for the **Execute** method. The enclosed example uses the GET URL approach for brevity. The same information could be provided to the same process via a POST with an XML document.

* **Input URL**:

<http://localhost/cgi-bin/pywps.cgi?&>

version=1.0.0&

service=WPS&

request=Execute&

identifier=MultiModelAvg&

responseform=responsedocument&

storeexecuteresponse=true&

status=true&

datainputs=[

Dataset=http://opendap.esgf/tasavg\_mm\_ECMWF\_200001-200012.nc,

<http://opendap.esgf/tasavg_mm_CFSR_200001-200012.nc>,

<http://opendap.esgf/tasavg_mm_MERRA_200001-2000012.nc>;

  Operation=avg;

Variable=tas;

Start\_date=20000101;

End\_date=20001231;

Max\_lat=50;

Max\_lon=-66;

Min\_lat=24;

Min\_lon=-125;

Start\_level=1;

End\_level=42;

Gridder=EMSF;

Grid=T85;

Output=OpendapURL;

OutputFileName=multi-model-avg-north-america-2000-ECMWF-CFSR-MERRA.nc;]

* **Sample Response Document**:

The initial feedback from the **Execute** operation is an XML response document similar to the one below. The **statusLocation** points to a separate XML status document, which contains the specific process details. Note that the status field indicates that the process has been accepted.

<wps:ExecuteResponse

xsi:schemaLocation=*"http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute\_response.xsd"*

service=*"WPS"* version=*"1.0.0"* xml:lang=*"en-CA"*

serviceInstance=*"http://nccs.gov/cgi-bin/wps?service=WPS&request=GetCapabilities&version=1.0.0"*

**statusLocation**=*"****http://localhost/wps/wpsoutputs/pywps-400f9a62-b6cb-11e4-bfbd-000c2930e06a.xml****"*>

<wps:Process wps:processVersion=*"1.0.0"*>

<ows:Identifier>MultiModelAvg</ows:Identifier>

<ows:Title>ESGF Use Case: Multi-Model Average</ows:Title>

<ows:Abstract>Averaging over multiple models across federated data.</ows:Abstract>

</wps:Process>

<wps:Status creationTime=*"2015-02-17T17:34:39Z"*>

<wps:**ProcessAccepted**>**Process MultiModelAvg accepted**</wps:**ProcessAccepted**>

</wps:Status>

</wps:ExecuteResponse>

Next is an intermediate status response document. Here the status field indicates that the process has started and that it is 40% complete.

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<wps:ExecuteResponse 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"*

xmln s:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute\_respo*

*nse.xsd"*

service=*"WPS"* version=*"1.0.0"* xml:lang=*"en-CA"*

serviceInstance=*"http://nccs.gov/cgi-bin/wps?service=WPS&amp;request=GetCapabilities&amp;version=1.0*

*.0"*

statusLocation=*"http://localhost/wps/wpsoutputs/pywps-f12686da-b6cb-11e4-880c-000c2930e06a.xml"*>

<wps:Process wps:processVersion=*"1.0.0"*>

<ows:Identifier>MultiModelAvg</ows:Identifier>

<ows:Title>ESGF Use Case: Multi-Model Average</ows:Title>

<ows:Abstract>Averaging over multiple models across federated data.</ows:Abstract>

</wps:Process>

<wps:Status creationTime=*"2015-02-17T17:39:37Z"*>

<wps:**ProcessStarted** **percentCompleted=40**">PyWPS Process MultiModelAvg successfully

calculated</wps:**ProcessStarted**>

</wps:Status>

<wps:ProcessOutputs>

</wps:ProcessOutputs>

</wps:ExecuteResponse>

The final status response document is shown below. Here the status field indicates that the process has succeeded and that the result is available for download at the corresponding opendapURL.

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<wps:ExecuteResponse 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"*

xmln s:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.opengis.net/wps/1.0.0 http://schemas.opengis.net/wps/1.0.0/wpsExecute\_respo*

*nse.xsd"*

service=*"WPS"* version=*"1.0.0"* xml:lang=*"en-CA"*

serviceInstance=*"http://nccs.gov/cgi-bin/wps?service=WPS&amp;request=GetCapabilities&amp;version=1.0*

*.0"*

statusLocation=*"http://localhost/wps/wpsoutputs/pywps-f12686da-b6cb-11e4-880c-000c2930e06a.xml"*>

<wps:Process wps:processVersion=*"1.0.0"*>

<ows:Identifier>MultiModelAvg</ows:Identifier>

<ows:Title>ESGF Use Case: Multi-Model Average</ows:Title>

<ows:Abstract>Averaging over multiple models across federated data.</ows:Abstract>

</wps:Process>

<wps:Status creationTime=*"2015-02-17T17:39:37Z"*>

<wps:**ProcessSucceeded**>PyWPS Process MultiModelAvg successfully

calculated</wps:**ProcessSucceeded**>

</wps:Status>

<wps:ProcessOutputs>

<wps:Output>

<ows:Identifier>**opendapURL**</ows:Identifier>

<ows:Title>opendapURL</ows:Title>

<wps:Data>

<wps:LiteralData dataType=*"integer"*>

**https://opendap.esgf/*f12686da-b6cb-11e4-880c-000c2930e06a*/multi-model-avg-north-america-2000-ECMWF-CFSR-MERRA.nc**

</wps:LiteralData>

</wps:Data>

</wps:Output>

</wps:ProcessOutputs></wps:ExecuteResponse>