MERRA/AS

The MERRA Analytic Services Project

Overview of Configuration, Deployment, and Testing for the Climate Data Services (CDS) Client Stack

John Schnase (PI), Dan Duffy (Co-I),
Glenn Tamkin, Denis Nadeau, Hoot Thompson,
Scott Sinno, and Mark McInerney

NASA Goddard Space Flight Center

Winter 2014

Code Drop Contents

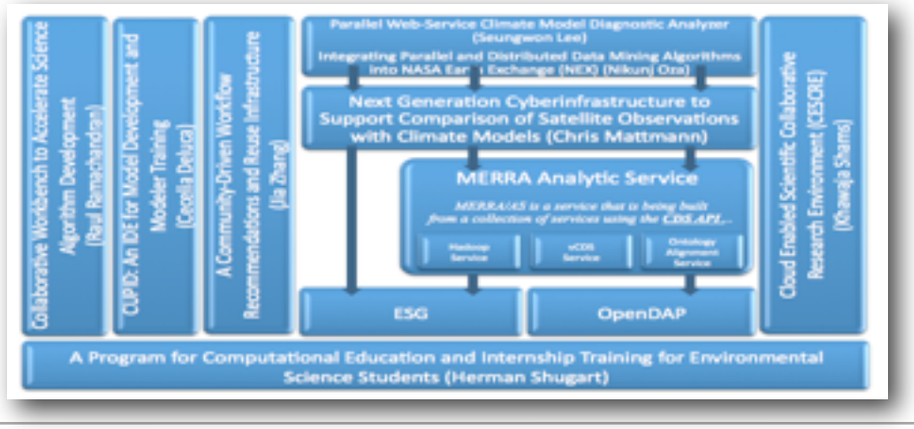
The MERRA Analytic Service (MERRA/AS) consists of client and server code. The server is a large code ecosystem customized to the MERRA/AS Hadoop cluster. This code drop is V0.8 of the MERRA/AS client distribution package, which is intended for use by our beta test community. V0.8 is feature complete, but as beta test software, this is the client's first exposure beyond the development organization – feedback on performance and problems would be much appreciated. The package includes the following:

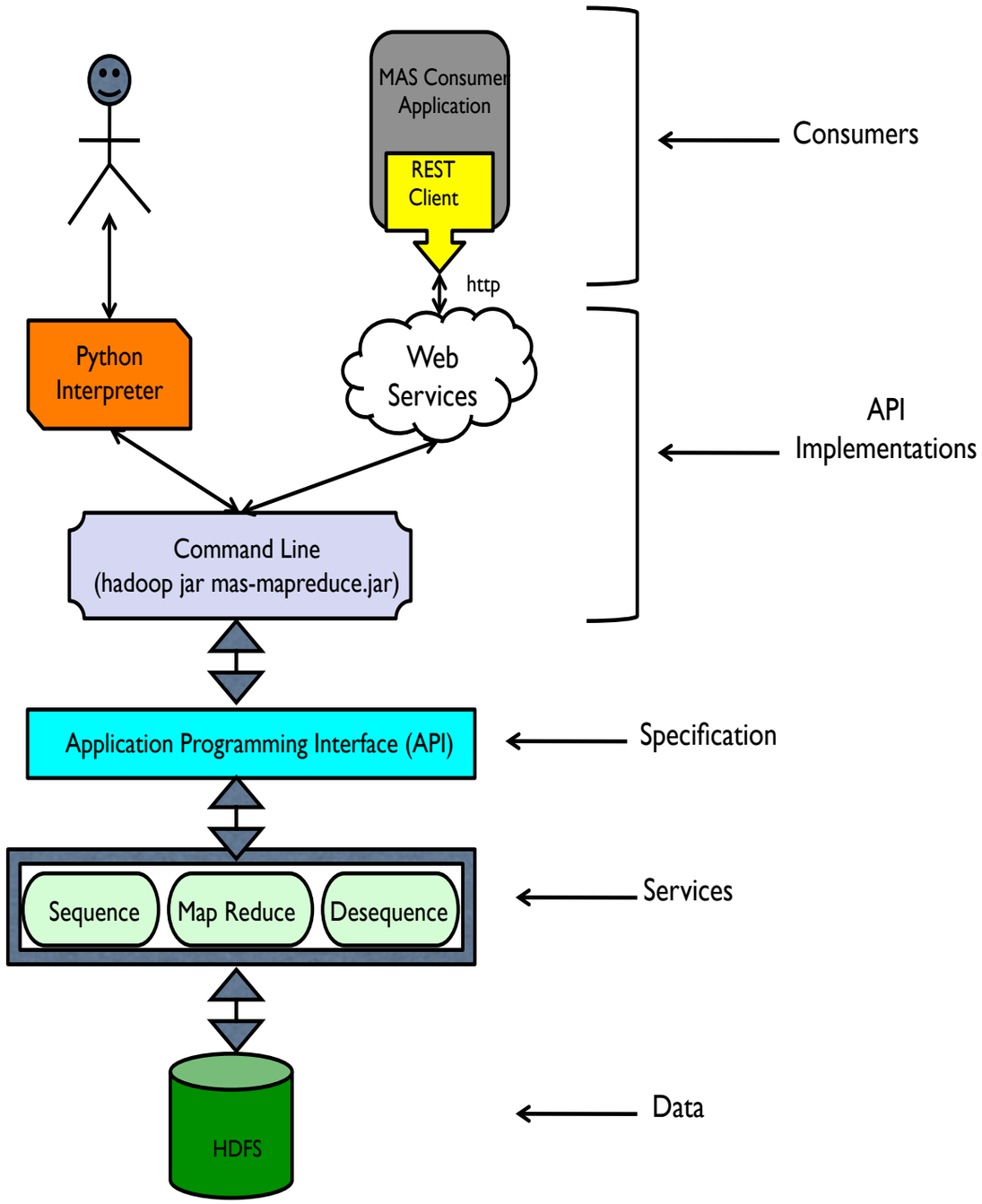
Source Code – The client distribution package contains all the source code necessary to build and execute a MERRA/AS application.

Build Files – The package contains all the instructions needed to run the MERRA/AS client along with any necessary tools such as Make, Ant, Maven, Python Buildout/Seuptools, etc. The client is Python, so no compilation is necessary.

A Simple End to End Use Case – We provide a sample use case, described below, and a more complex use case based on published research that uses MERRA data, which we refer to as the MERRA Wei Experiment.

Documentation – Finally, we provide documentation in the form of a README file (this file) summarizing the architecture and the MERRA/AS codeset organization and structure.





Source Code - Packaging and Installation

Packaging

- The distribution consists of a single tar ball called ‘cds_client_<version>.tar’
- This package contains Python source code for the Climate Data Services (CDS) Client stack along with Python applications and scripts to consume MERRA Analytic Services.
- The server is described in the MERRA/AS Project Interim Report, which is available on the ESTO website.
- The MERRA/AS Project Final Report will be available soon.

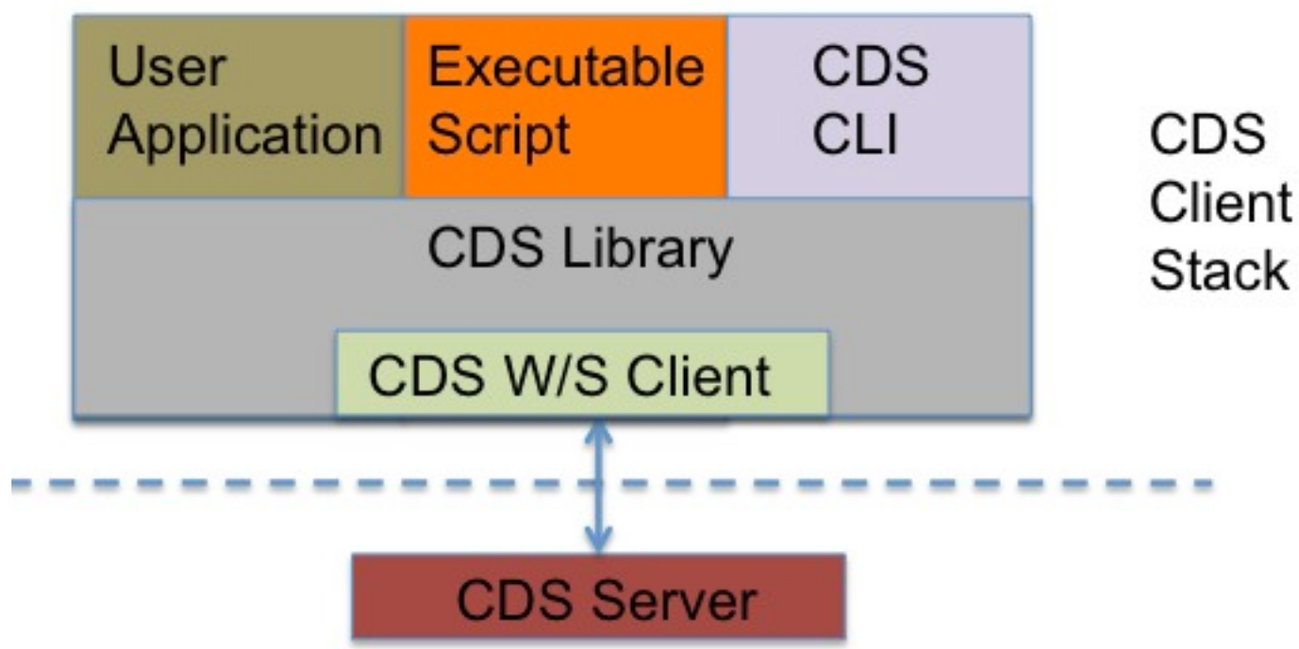
Installation

- Run the following command to decompress the tar:

```
$ tar -xvf cds_client_<version>.tar
```

- Navigate to the root directory for the README

```
$ cd python
```



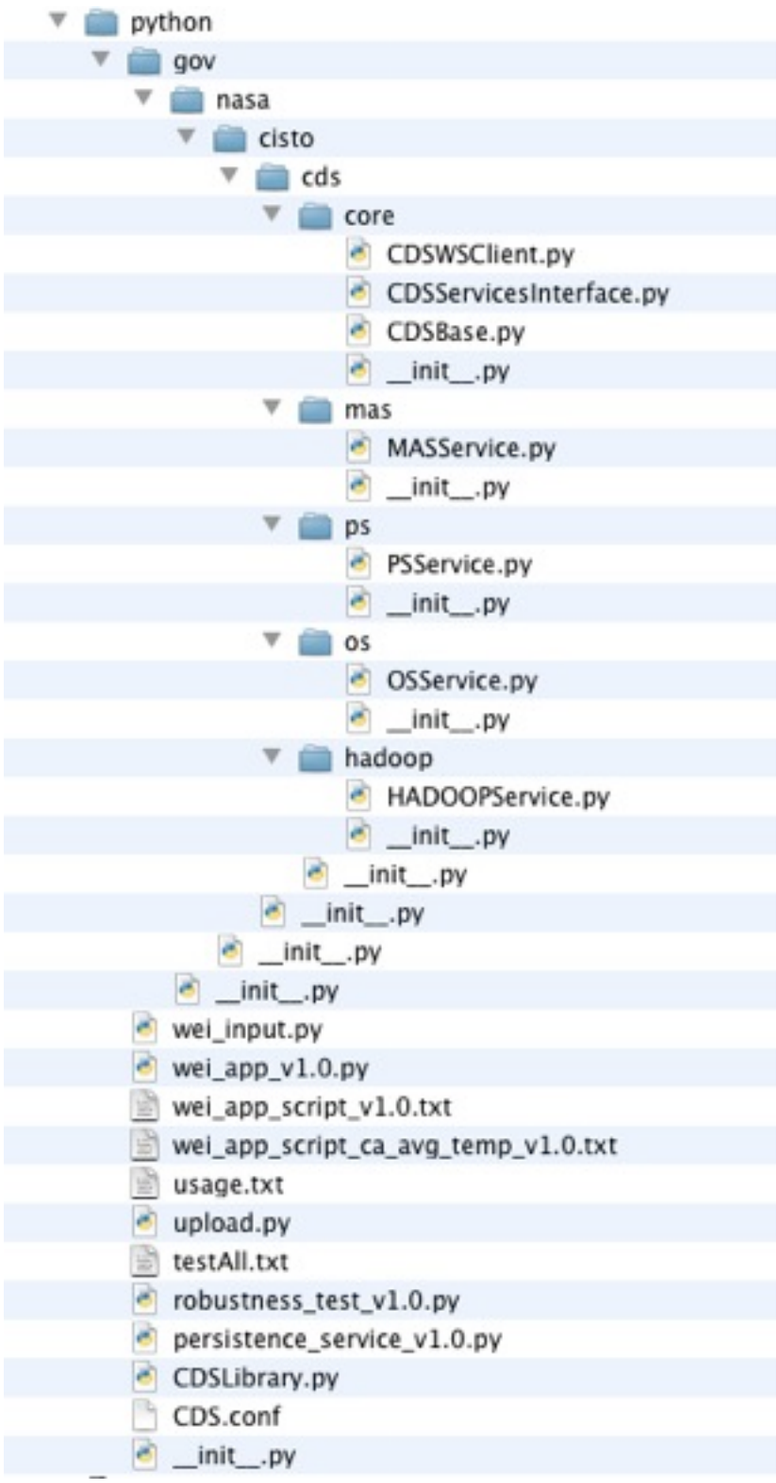
Source Code - Structure and Description

Structure

- The entire source code distribution is displayed in its natural hierarchy -->
- The CDS Client Stack application consists of a root Python application file (CDSLibrary.py) and configuration file (CDS.conf). The root file calls service and infrastructure classes that are packaged under gov.nasa.cisto.cds.
- ***See the "usage.txt" file for detailed instructions on how to execute the client.*** The remaining files (e.g., *v1.0) are test applications and scripts.

Description

- All files are annotated, but CDSLibrary.py contains the detailed description of the full CDS API. From an external programming or scripting perspective, this is the only file that the consumer needs to be aware of.
- *Service.py classes contain the specific implementation of the CDS API per domain. The externally available services include MASService.py (which contains the MapReduce primitives) and PSService.py (which contains persistence primitives).
- CDSServicesInterface.py defines the interface that is implemented by each service above
- When called, CDSLibrary.py passes the incoming parameters to the appropriate service (e.g., MAS, PS) which validates and formats the parameters and then requests an outbound web service call using CDSWSClient.py (which encapsulates the web service URL details)
- CDSBase.py provide supporting functionality like logging and config file access
- Note that CDSLibrary.py can be called from a Python program or a simple executable script (i.e., text file) containing a sequence of calls. Several examples are provided.



Build Files and Testing

Build Files (Make, Ant, Maven, Python Buildout/Setuptools, etc.)

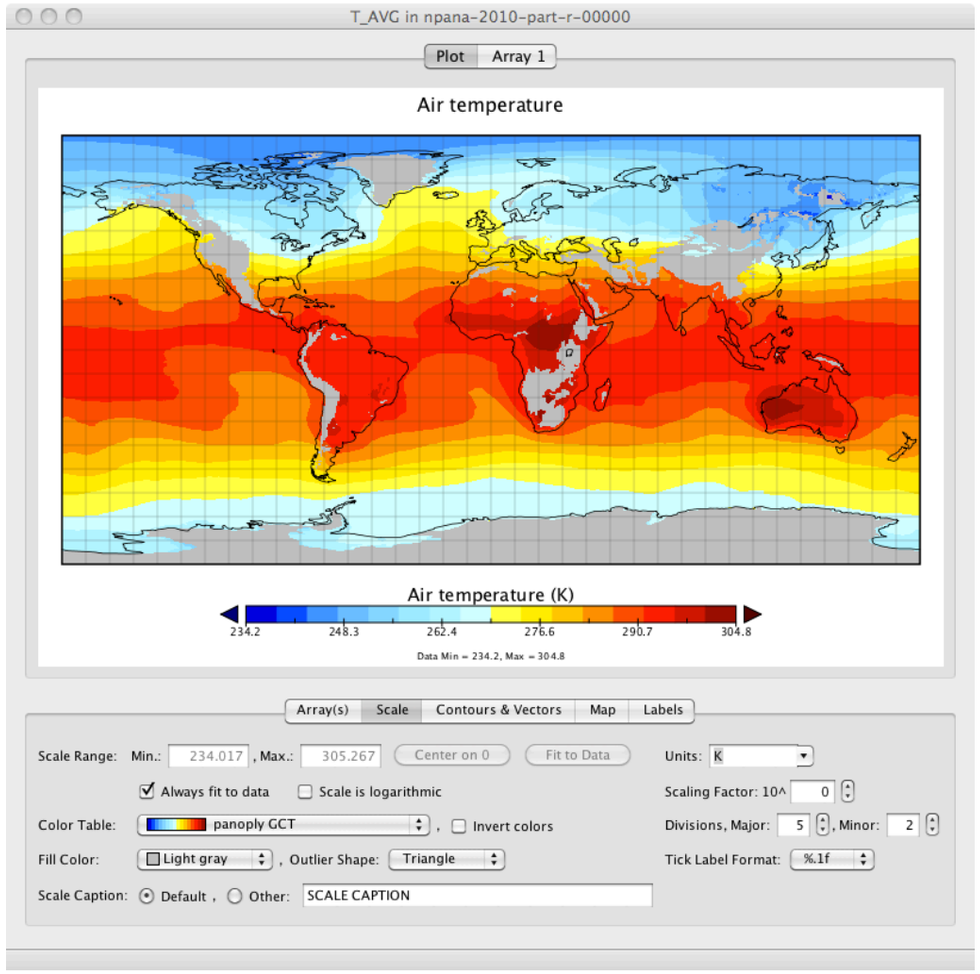
- The client is written in Python, so no explicit compilation is necessary
- Pre-requisites to execute the client include: 1) Python 2.6+, and 2) access to the MERRA/AS endpoint (i.e., `http://skyportal.sci.gsfc.nasa.gov)`. Access is currently limited to an authorized group of beta testers. To request access, please contact Mark McInerney at Mark.A.McInerney@NASA.gov.
- To run a Python application, type the executable followed by the application name:

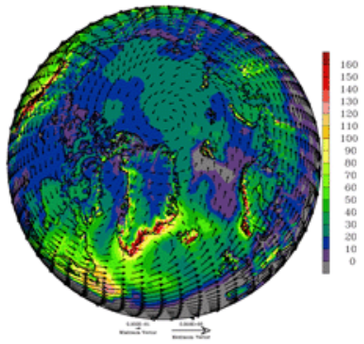
```
$ python wei_app_v1.0.py
```

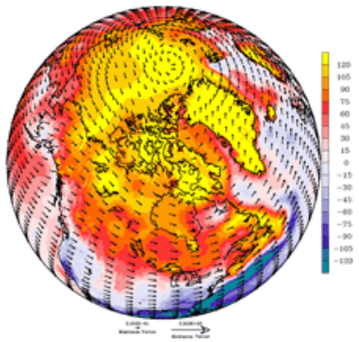
- To run an executable script, just run the executable:

```
$ ./wei_app_script_ca_avg_temp_v1.0.txt
```

- See the "usage.txt" file for runtime parameter descriptions for the enclosed tests
- The next slide illustrates sample use cases and the CDS API
- The CDS API is completed on the slide after that, which is followed by a etailed discussion of two simple end-to-end use cases







Sample Use Cases and CDS API Details (1)

Sample CDS Use Cases

- A data producer ***Executes()***s the Hadoop sequence service that generates sequence files
- A data producer ***Ingests()***s MERRA data, initiating the service that loads sequence files into HDFS
- A scientist calls ***Query()*** to determine which MERRA operations (e.g., average) and parameters (e.g., surface pressure, temperature, etc.) are available from the data
- A developer ***Ingests()***s the MapReduce code using the code hosting service
- A scientist ***Orders()***s seasonal average surface pressure for 2006-2009 with a time step of 3, initiating the MapReduce job
- A scientist monitors job progress using ***Status()*** until the job is complete
- A scientist ***Downloads()*** the results of the MapReduce job

CDS API Method Detail - Query

- The Query method executes a predefined service request that returns data in an XML instance package that conforms to a predefined standard schema
- Arguments
  - **service**: The name of the target service. For example: *MAS*
  - **request**: The name of the query to be processed. For example, *GetAvailableOperatorsByParameter* (queries supported by MAS Service)
  - **parameters:** 1 or more parameters of the format (name!value).
- Returns (resultSet or fault)
  - **resultSet**: query results in XML format ***- OR –***
  - **fault:** Error code (e.g., E_InvalidParameter)
- Example
  - `$ query(MAS, GetAvailableOperatorsByParameter, parameter!temperature)`
- Response

```
<temperature>
   <operator>average</operator>
   <operator>standard deviation</operator>
</temperature>
```

CDS API Method Detail - Ingest

- The Ingest method provides a generic way of providing submission information packages (SIP) to a service provider.
- Arguments
  - **service**: The name of the target service. For example: *HADOOP*
  - **operation**: An identifier that indicates the specific processing for the SIP. For example, *Put,* (operations supported by the HADOOP Service)
  - **SIP**: Submission information package (name!type!content)
  - **sessionId**: Optional session ID for the SIP if the operation is a result of a previous transaction (e.g., ingest parts of a single collection through multiple calls).
- Returns (sessionId!sessionStatus!sessionStatusDetail)
  - **sessionId**: Session id
  - **sessionStatus:** One word status of the session (e.g., Received, Completed, Failed)
  - **sessionStatusDetail**: Description of the session (e.g., "Ingest failed due to insufficient HDFS storage space")
- Example
  - `$ ingest(HADOOP, Put, MERRAMonth0109!MERRA!/home/gtamkin/m/01/09)`

SIP

CDS API Method Detail - Order

- The Order Method places an order for a predefined service request that does not return instantaneous results. Queries will be handled asynchronously with this method.
- Arguments
  - **service**: The name of the target service. For example: *MAS*
  - **request**: The name of the query to be processed. For example, *GetAverageTemperatureByTimeSpanAndStep* (orders supported by MAS Service)
  - **parameters:** 1 or more parameters of the format (name!value).
- Returns (sessionId!sessionStatus!sessionStatusDetail)
  - **sessionId**: session id. A unique UUID that is used to retrieve session status (e.g., f654c35c-f223-4787-a947-8787f532d3fe)
  - **sessionStatus:** One word status of the session (e.g., Received, Completed, Failed)
  - **sessionStatusDetail**: Description of the session (e.g., "Order failed due to lack of privileges")
- Example
  - `$ order(MAS, GetAverageTemperatureByTimeSpanAndStep, step!3, startDate!200202, endDate!200708)`

CDS API Details (2)

CDS API Method Detail - Download

- The Download method provides a means for retrieving packages associated with a sessionId from a service provider. In a typical asynchronous transaction, such as those associated with Order, this method obtains results from a service request.
- Arguments
  - **service**: The name of the target service. For example: *MAS*
  - **sessionId**: Session ID for the order.
  - **DIP**: Optional dissemination information package (name!type!content). When this parameter is empty, all packages associated with the supplied sessionId are returned. When a package name is specified within the DIP parameter, the service provider returns only the associated package.
- Returns
  - **DIP**: Requested dissemination information package
- Example
  - `$ download(MAS, f654c35c-f223-4787-a947-8787f532d3fe)`

CDS API Method Detail - Execute

- The Execute method implementation is optional for service providers. It is a service extension mechanism that allows a service provider to offer additional services that are accessed in a common way.
- Arguments
  - **service**: The name of the target service. For example: *iRODS*
  - **methodName**: The name of the method to be processed. For example, *ApplyDataRetentionPolicyToData* (methods supported by iRODS Service)
  - **parameters:** 1 or more parameters of the format (name!value).
- Returns (sessionId!sessionStatus!resultSet)
  - **sessionId**: session id.
  - **sessionStatus:** One word status of the session (e.g., Received, Completed, Failed)
  - **resultSet**: execute results in XML format
- Example
  - `$ execute(IRODS, ApplyDataRetentionPolicyToData, dataType!MERRA)`
- Response

```
f654c35c-f223-4787-a947-8787f532d3fe!Completed!<resultSet>327 files
   removed 4 TB freed</ResultSet>
```

CDS API Method Detail - Status

- Status is a method for session tracking. Once initiated, a session enters into different processing stages. The Status method offers the client a way of querying the current state of the session.
- Arguments
  - **sessionId**: Session ID.
- Returns (sessionId!sessionStatus!sessionStatusDetail)
  - **sessionId**: Session id
  - **sessionStatus:** One word status of the session (e.g., Processing, Completed, Failed)
  - **sessionStatusDetail**: Description of the session (e.g., "MapReduce job in progress")
- Example
  - `$ status(f654c35c-f223-4787-a947-8787f532d3fe)`
- Response

```
f654c35c-f223-4787-a947-8787f532d3fe!Processing!MapReduce job in
   progress
```

CLI Session – Query for Available Operations

- The scientist calls ***Query()*** to determine which MERRA operations (e.g., average) and parameters (e.g., surface pressure, temperature, etc.) are available from the data:

```
$ query(MAS, GetAvailableOperatorsByParameter, parameter!temperature)
$ <temperature>
$    <operator>average</operator>
$    <operator>standard deviation</operator>
$  </temperature>
```

Simple End-to-End Use Case (1)

With this use case, the consumer uses a browser to request MERRA/AS web services. An order is placed, the consumer polls until status is 'complete', and the consumer then downloads the results. [Note that this test does not exercise the CDS Client, but it does very that the MERRA/AS services are available.]

REST Service API (URL Conventions adhere to CDS API terminology):
http://<ip:port>/cds/<service>/<primitive>/<service_request>?<parameters> // order, query
http://<ip:port>/cds/<service>/<primitive>/<optional operation>?<parameters> // ingest, execute

The consumer Orders()s average temperature for time range [Jan. 2011-Dec. 2011] within Continental US for all vertical levels against NPANA monthly_means:

_____

http://skyportal.sci.gsfc.nasa.gov/cds/mas/order.php?GetAverageByVariable_TimeRange_SpatialExtent_VerticalExtent&variable_list=T&operation=avg&start_date=201101&end_date=201112&avg_period=12&min_lon=-125&min_lat=24&max_lon=-66&max_lat=50&start_level=1&end_level=42

```
<Response>
   <sessionId>h654c35c-f223-4787-a947-8787f532d3fe</sessionId>
   <sessionStatus>Received</sessionStatus>
   <sessionStatusDetail>Map Reduce job submitted at Wed Apr 17 13:17:06 EDT 2013</sessionStatusDetail>
</Response>
```

_____

http://skyportal.sci.gsfc.nasa.gov/cds/mas/status.php?session_id=h654c35c-f223-4787-a947-8787f532d3fe

```
<Response>
   <sessionId>h654c35c-f223-4787-a947-8787f532d3fe</sessionId>
   <sessionStatus>Completed</sessionStatus>
   <sessionStatusDetail>Map Reduce job completed successfully prior to Wed Apr 17 13:19:32 EDT 2013</
   sessionStatusDetail>
</Response>
```

_____

http://skyportal.sci.gsfc.nasa.gov/cds/mas/download.php?session_id=h654c35c-f223-4787-a947-8787f532d3fe

Response (File Download)

Simple End-to-End Use Case (2)

With this use case, the consumer uses the CDS Client to request MERRA/AS web services. Using the enclosed executable script, an order is placed, the consumer polls until status is 'complete', and then downloads the results.

The consumer Orders()s average temperature for time range [Jan. 2011-Dec. 2011] within Continental US for all vertical levels against NPANA monthly_means:

_____

```
$ more simple-end-to-end-use-case_v1.0.txt

#!/usr/bin/env python

import sys

from CDSLibrary import CDSApi

cds_lib = CDSApi()

service = "MAS"

north_american_parms =
    "&job_name=simple_end_to_end_use_case_na_avg_temp_2011-2012_instM_3d_ana_Np&collection=instM_3d_ana_
    Np&request=GetVariableBy_TimeRange_SpatialExtent_VerticalExtentvariable_list=T&operation=avg&start_d
    ate=201101&end_date=201102&avg_period=2&min_lon=-125&min_lat=24&max_lon=-66&max_lat=50&start_level=1
    &end_level=42"

class UserApp(object):

    if __name__ == '__main__':

         # exercise all canonical operations

         cds_lib.avg(service, north_american_parms, "/tmp/
    simple_end_to_end_use_case_na_avg_temp_2011-2012_instM_3d_ana_Np.nc")

$./simple-end-to-end-use-case_v1.0.txt
```

MERRA/AS Parameters

Methods

- The methods (i.e., web services primitives) that are supported by the CDS API were described earlier in this document. The hard specification of this abstraction is included in CDSLibrary.py
- All methods require a specific signature. However, some (i.e., order(), execute()) allow dynamic behavior by accepting a string of RESTful key-value pairs (e.g., `&start_date=201101&end_date=201102)` that can override the default behavior of the method. If the key appears more than once in the list, the last one encountered is used by the application.
- Using this approach, CDSWSClient.py basically formats the incoming parameters accordingly and performs the web service call, passing the RESTful key-value pair list.

Parameters

RESTful Key	Description	Sample Value(s)	Units
request	Service request	GetVariableByOperation	String
collection	MERRA collection to process over	instM_3d_ana_Np	String
start_date	The starting date to process from	20110201 (YYYYMMDD)	String
end_date	The ending date to process to	20110630 (YYYYMMDD)	String
job_name	Name of Hadoop job	Average_temp_ca	String
variable_list	The variable of interest	T, H, V, PS…	String
operation	The operation to perform	avg,min,max,sum,count,var	String
min_lon	Minimum longitude - degrees_east	-180.0	double
min_lat	Minimum latitude - degrees_north	-90.0	double
max_lon	Maximum longitude - degrees_east	180.0	double
max_lat	Maximum latitude - degrees_north	90.0	double
start_level	Start vertical level	1	double
end_level	End vertical level	42	double