EUROPEAN COMMISSION

Community research

is-enes

INFRASTRUCTURE FOR THE EUROPEAN NETWORK FOR EARTH SYSTEM MODELLING

IS-ENES2 DELIVERABLE (D -N°: 5.4)

ENES Data Service Infrastructure requirements

Original title: ***Report on metadata controlled vocabulary extensions***

File name: {IS-ENES2_D5.4.pdf}

Reporting period: **01/04/2016 – 31/03/2017**

Author(s): **Martin Juckes,**

Grigory Nikulin

Reviewers*:* **Lars Bärring**

Poppy Townsend

Release date for review: **24/03/2017**

Final date of issue: **13/04/2017**

Revision table

Versio n	Date	Name	Comments
0.5	24/3/2017	Draft	First version before review
1.0	11/4/2017	Revised	Revisions based on reviews.

Abstract

The Infrastructure Support for the European Network of Earth System Modelling for climate (IS-ENES2) project has contributed to a range of vocabularies used for managing climate data. The vocabularies facilitate efficient management and dissemination of climate data. This work complements work done in WP11 on meta-data services and reported in D11.4. The vocabularies described here are designed for use in file meta-data: in the naming of files and the internal meta-data.

Project co-funded by the European Commission's Seventh Framework Programme (FP7; 2007-2013) under the grant agreement n°312979

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants including the Commission Services	
RE	Restricted to a group specified by the partners of the **IS-ENES2** project	
CO	Confidential, only for partners of the **IS-ENES2** project	

Table of contents

Executive Summary

Work Package 5, Data Networking, has contributed to two important areas of vocabulary development, which will be detailed below. The work supports the Climate Model Inter-comparison Project, Phase 6 (CMIP6) of the Working Group on Coupled Models (WGCM) and the Coordinated Regional Climate Downscaling Experiment (CORDEX) of the Working Group on Regional Climate (WGRC), both within the portfolio of the World Climate Research Programme (WCRP).

This report describes work done primarily at STFC and SMHI over the duration of the IS-ENES2 project. For the Climate Model Inter-comparison Project, Phase 6 (CMIP6) this work delivered the Data Request, while for the CORDEX the primary outcome was the Data Reference Syntax. The CMIP6 Data Request consolidates requirements from 21 independent Model Inter-comparison Projects (MIPs) which have been endorsed by the Working Group on Coupled Models (WGCM). The CORDEX Data Reference Syntax specifies the file naming and metadata requirements for the WGCM CORDEX archive.

These data standards enable automated data management and reliable data analysis. Scientists will be presented with data which has originated in dozens of different computing centres around the world, but should find common formats with a high level of detail about many hundred diagnostics which are generated for the archive.

1. Preamble

This document summarises the work within IS-ENES2 to support the CMIP6 data request and CORDEX in terms of information management. For more information, see w3id.org/cmip6dr for the CMIP6 data request and is-enes-data.github.io/ for the CORDEX vocabularies.

The IS-ENES Data Services are intended to deal with large volumes of complex data products from many sources. Efficient management of these data products, and effective communication between users, service managers and data providers relies on an evolving set of data standards. The IS-ENES community has long played a leading role in this activity, and work funded by IS-ENES2 has made a substantial contribution.

2. CMIP6 Data Request

2.1 Data Request XML format

The Data Request defines the diagnostics which are requested from modelling centres participating in the sixth phase of the Climate Model Inter-comparison Project (CMIP6).

The Data Request is presented as two XML files: a configuration file and the content. Each file has an associated XSD schema. The XSD schema for the content file is generated automatically from the configuration file. For many users it will be more convenient to deal with the python interface or web and spreadsheet versions of the request, which will be described in a separate document. The transformation to an XML format from the traditional spreadsheet format is designed to deal with a number of issues associated with growing complexity and a need to support automation driven by the scale of the request. In order to preserve continuity, many of the records in the XML files will have a direct relation to spreadsheet rows in the traditional format.

A separate document describes a simple python API for the data request.

The variables are now also listed in a spreadsheet of MIP tables:

proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/CMIP6_MIP_tables.xlsx

In release 01.beta.27 a supplement was added, containing records on quality control and (under development) information about physical relationships between variables. The supplement is presented as a separate file in order to simplify the version management of the main request document.

2.1.1 Objectives

The broad objectives of the data request are:

(1) Define variables, together with technical information required for generation of output files;

(2) Define collections of variables, from specified experiments, which are needed for or relevant to specific scientific objectives;

2.1.2 Files

The framework schema:

http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/vocabFrameworkSchema_01beta.xsd

Configuration file:

http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/dreq2Defn.xml

Data request schema:

http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/dreq2Schema.xsd

Data request XML:

http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/dreq.xml

Supplement schema:

http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/dreqSuppSchema.xsd

Supplement:

http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest/dreqPy/docs/dreqSupp.xml

2.1.3 Overview

Configuration file

The XML Data Request is presented as a configuration file and a content file.

The configuration file contains three types of information:
(1) Layout information which is used to generate the content schema;
(2) Comments on the purpose and intent of attributes;
(3) Technical labels to facilitate automated navigation of the contents.

If users wish to exploit the XML files directly it is recommended that they make use of the configuration file, as the information types (2) and (3) are not embedded in the content file.

The document is split into sections, each of which contains a list of records with a common syntax and purpose. Each section could be considered as a table in a relational database. Each section of the document is defined by a “table” element with the following attributes:

- label (e.g. 'var'): a name for a section of the content – will be used as the XML element name;
- title (e.g. 'MIP variable”): a longer, human readable string;
- id: an opaque name;
- itemLabelMode: specifies whether the “label” attribute of records in this section should permit use of '-';
- level: an integer, designed to assisted automated processing by giving an indication of the structure of the request;
- maxOccurs: maximum number of times the section is allowed;
- labUnique [Yes|No]: set to yes if label values for records are unique within each section.

Within each section there are definitions for attributes of items. Each item attribute is defined using the following configuration attributes:

- label: this will be the attribute name;

- title: a longer string explaining usage;
- class: the class supports automation. e.g. attributes which refer to another record in the document will have the class set to “internalLink”;
- type: the xsd content type (e.g. “XS:STRING”);
- techNote: to support automation. e.g. if class is “internalLink”, this attribute should be set to the name of the intended section:
- required: indicates whether the attribute is required;[1]
- usage: notes on the usage of the attribute.[1]

Since 01.beta.33 all properties (title, valid_max, valid_min, etc) in all sections have been given a valid title to aid legibility of the document.

In addition to the standard XSD content types “string”, “boolean”, “integer”, “duration” and “float”, the following types are defined:

- st__integerList: a list of integers;[1]
- st__integerListMonInc: a monotonic increasing list of integers (monotonicity is not checked by the XSD schema, but is verified by the python API);[1]
- st__floatList: a list of integers;[1]
- st__stringList: a list of words;[2]
- st__attLabel_def: a string composed of characters “a” to “z”, “A” to “Z”, “0” to “9” and “-”;

The following table summarises the specifications of the core attributes:

label	**title**	**description**	**usage**
label	Record Label	A single word, with restricted character set	A short mnemonic word which is potentially meaningful but also concise and suitable for use in a programming environment
uid	Record Identifier	Unique identifier	Must be unique in the data request. For well known concepts this may be related to the label, but for items such as simple links between concepts a random string will be used.
title	Record Title	A few words describing the object	A short phrase, suitable for use as a section heading
description	Record Description	An extended description of the object/concept.	

1 New in 01.beta.17

2 New in 01.beta.19

useClass	Record Class	The class: value should be from a defined vocabulary. All records in the schema definition section must have class set to "__core__".	The useClass declared for an attribute can affect its interpretation in the Python package. For example, attributes labelled as "useClass=internalLink" should refer to another data request record.
type	Record Type	The type specifies the XSD value type constraint, e.g. xs:string.	Used in the dreqPy package to determine data type.
techNote	Technical Note	Additional technical information which can be used to specify additional properties.	
superclass	Superclass	States what class the property is derived from	
id	Alternative identifier	Alternative identifier	For sections, the id provides a short alias for the section label.
itemLabel Mode	Item Label Mode	Item Label Mode	
level	Level	Level	Redundant
maxOccurs	Maximum number of permissible occurrences of this section	Maximum number of permissible occurrences of this section	Used in defining sections. In the CMIP6 Data Request each section only occurs once.
labUnique	Set true if label of each record is unique within section	Set true if label of each record is unique within section	Used in defining sections.
usage	Usage notes	Notes on the usage of the predicate/concept defined by this node	

The above attributes provide the framework for detailed description of data request attributes and diagnostics.

Content file (dreq.xml)

The content file contains three elements at the top level: “prologue”, “main” and “annex”[3]. The “prologue” contains Dublin Core metadata describing the document and a PAV version attribute holding the document version[4]. The “main” element has the sections specified in the configuration file, and within each section a list of records called “item” elements. Each item element has attributes as specified in the configuration file, a different set of attributes for each section. There are no child elements or text content, all the information is in the defined attributes. This flat document structure is designed to make facilitate easy document parsing. Every item, across all sections, will have at least these 3 common attributes which are intended to give basic information about the item, thus enabling standardisation in error tracking:

- uid: an identifier which is unique within the document;
- label: a short name, using only the characters a-z, A-Z, 0-9 and '-' (in some sections the '-' is disallowed);
- title: a longer name.

The “annex” element also contains a list of sections with the same structure as in the “main” element. The “annex” has been introduced to allow some flexibility in the version management.

Sections

The data request document is divided into 26 sections, 6 of which contain information about variables, output format and their priorities. An index to the request sections is available here: http://clipc-services.ceda.ac.uk/dreq/index.html .

The sections, with section numbers, are listed below:

1.1 Model Intercomparison Project [mip]

1.2 MIP Variable [var]

Each MIP variable record defines a MIP variable name, associated with a CF Standard Name.

1.3 CMOR Variable [CMORvar]

Each Output variable record corresponds to a MIP table variable specification. In a change from the August draft, this record does not contain the “priority” attribute: the priority is now set in the “Request Variable” record. The other change is that a collection of attributes specifying dimensions have been moved into the “structure” record, and each CMOR variable record links to one structure record. This will facilitate provision of clear and consistent definitions of output formats.

1.4 Request variable (carrying priority and link to group) [requestVar]

The request variable is now a short record which combines a CMOR variable with a priority and assigns it to a request group. The request variable records define the contents of each request group.

3 New in 01.beta.16

4 New in 01.beta.29 (purl.org/pav/2.3)

1.5 Experiments [experiment]

The experiment record contains the key information from the “Experiment” sheet of the request template, including the tier of the experiment, the duration and start/end dates.

1.6 Scientific objectives [objective]

Each request for data is associated with at least one of the scientific objectives listed by the MIPs. Data providers may wish to support a limited number of scientific objectives (perhaps because their model is not considered appropriate or because resources are limited) and can filter the list of requested diagnostics accordingly.

1.7 Specification of dimensions [grids]

A section for the CMOR dimensions specifies the structure of the axes of the requested diagnostics.

1.8 CF Standard Names [standardname]

The reference list of CF standard names is provided at cfconventions.org, but the definitions of terms used in the data request are copied into this section so that the detailed definitions are easily accessible to data request users.

1.9 Experiment Group [exptgroup]

The experiment group defines a collection of experiments within a MIP which might be part of a collective data request.

2.1 Spatial dimensions [spatialShape]

The spatial shape record contains the spatial dimensions of the field, and also, for convenience, an integer specifying the number of levels if that number is specified. A boolean level flag is set to “true” if the number of vertical levels is specified.

2.2 Temporal dimension [temporalShape]

The temporal shape record contains the temporal dimensions.

2.3 Dimensions and related information [structure]

The structure record combines specification of dimensions, cell_measures and cell_methods attributes. Spatial and temporal dimensions are specified through links to “spatialshape” and “temporalshape” records.

3.1 Request variable group: a collection of request variables [requestVarGroup]

The request variable groups collect variables.

3.2 Request Item: specifying the number of years for an experiment [requestItem]

The request item links a collection of variables with a specific experiment or group of experiments, and a temporal range for output. The “esid” attribute links to an experiment, and experiment group or a MIP. In the latter case, the request applies to all experiments defined by that MIP. The Request Item includes a “Tier Reset” attribute (“treset”)[5] which can override the Tier assigned to the experiments identified by “esid”. Has an optional link to a time slice[3].

5 New in 01.beta.17

3.3 Request link: linking a set of variables and a set of experiments [requestLink]

The request link records specify some additional information about variable groups, concerning shared output requirements and objectives.

3.4 CMOR Table Sections [tableSection]

3.5 Model configuration options [modelConfig]

3.6 Links a variable to a choice element [varChoiceLinkC]

Presence of a link indicates that there is a choice of different representations for a diagnostic.

3.7 Link between scientific objectives and requests [objectiveLink]

Each objective link record joins one objective to one request link. Some requests are linked to multiple objectives and most objectives are linked to multiple requests.

3.8 Remarks about other items [remarks]

The remarks section contains additional comments about other records. It can be used to add detail without adding to the complexity of the other sections.

3.9 Links a variable to a choice element [varChoiceLinkR]

Indicates that there is a ranked choice of variables, and that only one of the ranked list is required.

3.10 Indicates variables for which a there is a range of potential CMOR Varibles [varChoice]

There are several instances where variables defined in the tables are mutually exclusive options of which only one should be requested. The varChoice section is designed to hold this information, but is not yet complete. Examples are between ocean cell volume on a fixed grid for some models and monthly means for others, or between 6 hourly pressure level data on 8 levels vs. 4 levels for different objectives in the HighResMIP request.

3.11 Time Slices for Output Requests [timeSlice]

Specifies time slices (i.e. subsets of an experiment when data for the full duration of the experiment is not required.

Chapters 4 to 6: no content at present.

6.1 Tags

Tags related to processing requirements associated with some diagnostics to aid automated processing.

6.2 Relations between CMOR variables [varRelations][6]

Provides structured information about the difference between variables of the same name and frequency in different tables. E.g. different masking, temporal mean vs. point, different vertical structure (model levels vs. pressure levels).

6 There appear to be a number of broken links in this area .. the use of these records is under development.

6.3 Variable relation link [varRelLnk]

Provides links between CMOR variables and varRelation records, expressing relationships between variables.

Quality Control Ranges [in supplement]

Extends the information provided in the valid_min, valid_max, ok_mean_min_abs, ok_mean_max_abs attributes which were present in the CMIP5 CMOR tables. In this section there are also attributes valid_max_status etc which indicate the level of confidence in the suggested limits:

- **robust**: A well characterised limit based on a rigorous constraint (e.g. and area fraction must be between 0 and 1) or on a large ensemble of consistent model results.
- **suggested**: A limit which may not be reliable, but which is based on a range of models or plausible arguments.
- **tentative**: Very limited information – e.g. only one or two models in CMIP5 provided the parameter.

Further discussion is available in a draft document on Quality Control range[7], and web pages presenting a review of CMIP5 ranges shows the information being used to construct the control values[8].

X.1 Core Attributes [__core__]

X.2 Data Request Attributes [__main__]

X.3 Section Attributes [__sect__]

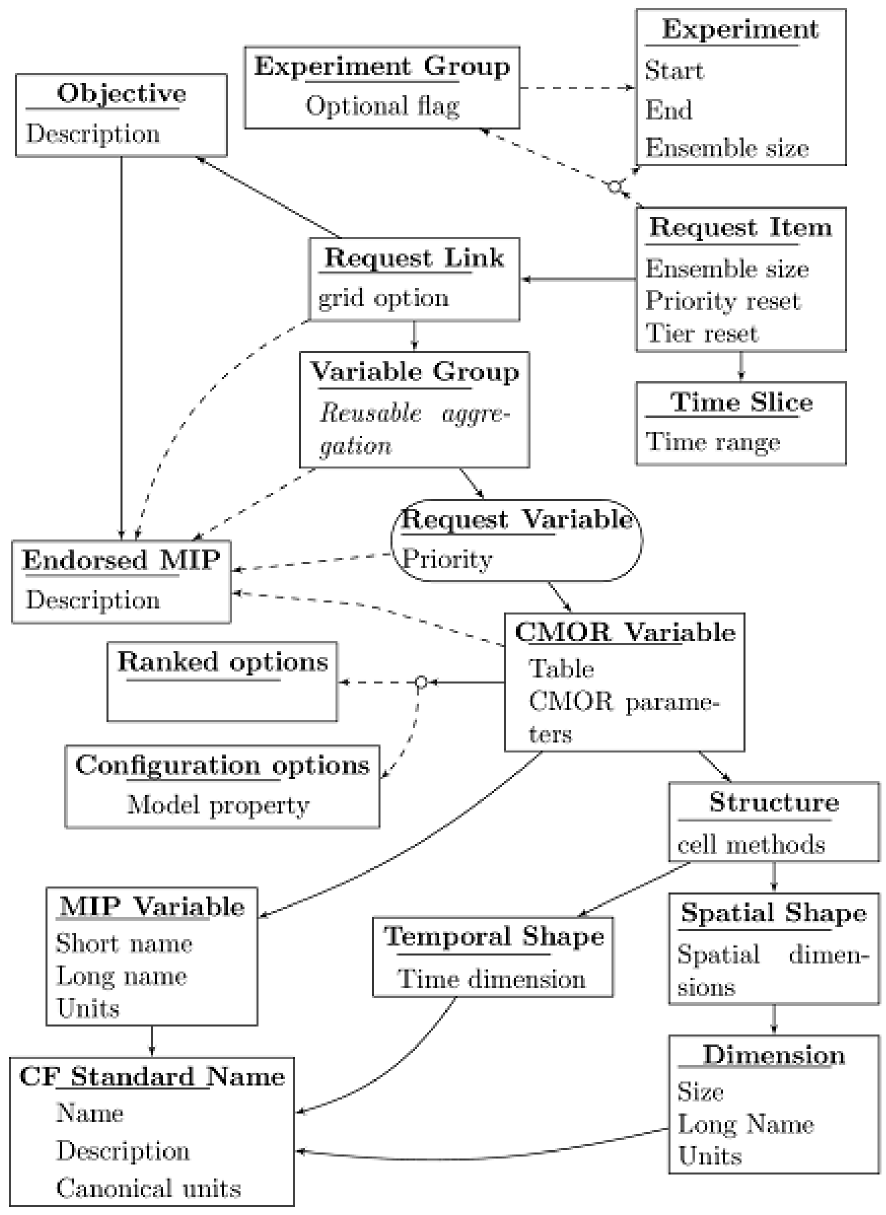
Defines the attributes which are used to describe each section.

2.1.4 Diagrammatic view of Data Request sections

7 https://docs.google.com/document/d/1cvSphy3Hb07t92BJvtqEBM9DMbsOSdENbwLJxw4AmH8/

8 http://clipc-services.ceda.ac.uk/ranges/ or http://w3id.org/cmip6dr/ranges/day_clt.html for a direct link to a single variable.

The following diagram illustrates the links between the different sections.



Linkage between data request elements. The Endorsed MIPs link to several different records because of the multiple roles they play in creating the request. CMOR variables, for example, link to the MIP which is responsible for defining the variable. The complete list of variables requested by a MIP, which typically includes many variables defined in CMIP5, is obtained by following the request links associated with that MIP, filtered by experiment(s), priority and objective if you are interested in a selective list. Dashed lines indicate links which are optional or supplementary. Solid lines indicate the primary links needed to decipher the request.

.

2.1.5 Discussion

The layout of the variable definitions has been rationalised into 5 sections: the “MIP variables” defining the physical parameters, “structure”, “spatialShape” and “temporalShape” defining output configuration and a “CMOR Variable” bringing all these together. The Request Variable table then links CMOR variables together in Request Groups. The request groups give the MIP coordinators the ability to pick and choose precisely the variables needed for each analysis, avoiding requests for unnecessary data. This will result in request groups which contain overlapping data requirements. The use of links back to CMOR variables make it possible to unambiguously determine the union of any set of request groups (provided that there is no duplication of variable in the CMOR variables section).

The structured layout ensures that the many hundreds of diagnostics defined in the request can be dealt with in a consistent manner.

2.2 Data Request Python API

2.2.1 Introduction

The Data Request is presented as two XML files whose schema is described in a separate document. A python module is provided to facilitate use of the Data Request. Some users may prefer to work directly with the XML file or with spreadsheets and web page views, but this software provides some support for those who want to use a programming approach.

Objectives

To provide intuitive access to the complete collection of information held in the data request document.

Overview

The basic module provides two objects, the first of which contains the full information content. The 2nd provides some indexing arrays to facilitate navigation through the request.

2.2.2 Installation

PyPi repository

The package is available from the test python repository at https://pypi.python.org/pypi/dreqPy/01.00.00

To install as user (watch the command response to see where pip places the package):

or, with administrator privileges:

```
pip install -i https://pypi.python.org/pypi dreqPy==01.00.00
```

Download code from subversion

The code is kept in a subversion repository, with a tag for each release.

```
svn co  http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/01.00.00
# or: svn co  http://proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/tags/latest
cd dreqPy
python simpleCheck.py
```

After extracting the code, the script "simpleCheck.py" will run a few basic checks and report any problems.

Requirements

The following python versions are supported: **python 2.6.6, 2.7 or 3.x.**

The core modules of the package only uses core python modules:

xml, string, re, collections, shelve, sys, os

The software runs significantly faster in python 3.x.

In the command line tool there is a dependency on xlsxwriter, which is used to write tables of variables.

Getting started

Version

To print the version number:

```
drq -v
```

Help

To print help text:

```
drq -h
```

```
drq -unitTest
```

Tests

To run some test, checking a range of package modules.

2.2.3 Command line interface

Once the package is installed, it can be invoked with the "drq" command. It can also be invoked from the directory containing the source code using "python dreqCmdl.py <args>", using the same arguments as for "drq", as described below.

Obtaining a list of variables requested by a MIP

```
drq -m HighResMIP -t 1 -p 1 --printVars --printLinesMax 20 -e historical
--xls
```

This command will provide a list of variables requested by HigResMIP for the CMIP6 historical experiment, and provide them in a spreadsheet in the "xls" directory. If the shorter form "drq -m HighResMIP" is used, the software will print out a volume estimate and not provide the list of variables.

The “-t” and “-p” arguments refer to the tier of experiments and priority of variables, described in section 4.3 below. If omitted, they both default to unity.

The “--printVars” and “--printLinesMax” arguments can be used to generate a print-out of the largest variables by estimated volume. This is only intended as a summary. More complete information is provided in the spreadsheets.

The “-e” argument takes the name of an experiment, or of an endorsed MIP, or “CMIP”. If an endorsed MIP name is given, the code will provide results for experiments defined by that MIP. For example, “drq -m C4MIP -e ScenarioMIP” provides information on the data requested by C4MIP from experiments defined by ScenarioMIP. If “-e CMIP” is used, the information is provided for the DECK and CMIP6 historical experiments.

By default, the output includes the data requested specifically be the endorsed MIP identified in the argument and the core request. To omit the latter, see section 4.2.

The Core Request, Multiple MIPs and Selection by Objective

To obtain information about multiple MIPs, simply list them, separated by commas:

By default, the output includes the core request which is requested from all models participating in

```
drq -m C4MIP,LS3MIP,LUMIP -t 1 -p 1 --printVars --printLinesMax 20 -e
historical --xls
```

CMIP6. To omit this, add the argument “--omitCmip”.

All data requested is associated with specific scientific objectives. In some cases the MIPs have specified multiple objectives with different data requirements and modelling groups may elect to provide data only for a selection of objectives.

The command in the box below will produce a list of variables required to support the RFMIP “Rapid Adjustment” and “Aerosol Instantaneous Forcing” objectives and the HighResMIP “Ocean” objective.

```
drq -m  RFMIP:RapidAdjustment.AerosolIrf,HighResMIP:Ocean --xls
```

This gives a data volume estimate of 3Tb, compared to 30Tb if all objectives of HighResMIP and RFMIP are supported.

Selection by Tier of Experiments and Priority of Variables

Within the request, each experiment is assigned a Tier, and requests for variables all have priorities. In CMIP5, a priority was assigned to each variable, but in CMIP6 variables may have different significance for different MIPs, so the priority is assigned for each request of a variable.

There is also a dependency of the variables requested on the tier: LS3MIP has a request for 3-hourly data which should cover the whole length of the historical simulation if their tier 2 experiments are being performed, but only a limited time slice if only tier 1 is being performed.

Intersection of request

An option has been added to support the evaluation of the intersection of requests:

```
drq --intersection -m HighResMIP,DynVar  --printVars
```

The following will evaluate the intersection of the HighResMIP and DynVar requests.

Specifying model grid sizes for volume estimation

The package uses a set of default model grid sizes to estimate the data volumes:

Label	Description	Default
nho	Number of horizontal grid cells in ocean grid	259200
nlo	Number of vertical levels in ocean grid	60
nha	Number of horizontal grid cells in atmosphere grid (also used for land surface fields).	64800
nla	Number of vertical levels in atmosphere grid	40
nlas	Number of vertical levels in the stratosphere (used for a small set of variables)	20
nls	Number of vertical levels in the soil model	5
nh1	Number of latitudes (used for transects and zonal means in both atmosphere and ocean at present, so as to give these fields a non-zero volume).	100

Table 1: Model configuration

The first four of these, specifying the size of the ocean and atmosphere grids, are the most relevant for volume estimation, since they determine the shape and dimension of the largest requested diagnostics. The last 3 have little impact. For example, the following command resets the number of vertical levels in the atmosphere to 45.

```
drq -m HighResMIP,C4MIP --mcfg 259200,60,64800,45,20,5,100
```

Legacy volume estimation

```
drq -m HighResMIP,C4MIP --legacy --xls -p 2
```

Earlier versions of the code used a different set of internal routines to generate volume estimates and variable lists. There were some problems dealing with the logic of grid selection, so the code was reconfigured. The old routines can still be accessed using the command above.

Grid Policy

The phrase “Grid Policy” is used here to refer to the decisions taken when there are a number of possible options regarding the grid used to store global fields. Prior to 01.beta.37 there were no options in the command line interface. Now there are two parameters which can be set:

--grdpol: “native”, “1deg” (default) or “2deg”: the type of grid to be used for ocean data when no preference has been expressed by any of the MIPs requesting the data.

--allgrd: (on if flag present, off by default): if on, all requested grids for each variable are included. if off, only the highest resolution data (assuming “native” is higher than “1deg”) is preserved.

Setting “--grdpol native” results in a significant increase in data volume relative to the new default. These options only take effect when the “--sf” flag is set (see above).

The grid policy options may be adjusted in the future to reflect priorities set by the WIP, meanwhile, these options make it possible to explore the consequences of various choices.

Spreadsheet formats

The data request tool produces two categories of spreadsheets:

- variable lists: files with names starting "cmv";
- volume estimate workings: files with names starting "requestVol".

File names are of the form <stem>_<id1>_<id2>_<tier>_<priority>[_<sfx>].xlsx

- stem: "cmvmm" (variable list for experiments requested by a MIP), "cmvme" (variable list for experiment), "cmvume" (variable list which are requested by one MIP but not by others). These are extended with "fr" if the option "--xfr" is used to generate sheets sorted by frequency and realm;
- id1: the requesting MIP or, if more than one MIP is involved, a string formed using the first two letters of each MIP, e.g. "cm.ls.lu" for CMIP, LS3MIP and LUMIP;
- id2: the experiment or MIP specifying a group or experiments;
- tier: the maximum tier set in the query;
- priority: the maximum priority set in the query;
- sfx: the suffix is added if variations on the default grid selection option are used: "fn" if "--grdforce native" is used (all data to be put on native grid), "f1" if "--grdforce 1deg", and "dn" if "--grdpol native" is used (use preference expressed by requesting MIPs, when present, otherwise use the native grid). The default is to use the preference expressed by requesting MIPs, when present, otherwise use 1 degree grid.

Variable lists

The variable lists are modelled on the CMIP5 standard output spreadsheet. If the command is invoked for multiple MIPs and experiments, there will be a file for the aggregated list of variables, as well as files for each individual MIP and experiment.

Additional columns are added to list the MIPs requesting each variable, and the MIPs defining the experiment each variable is requested for (these columns are more useful in the aggregated files – they contain little useful information in files corresponding to a single MIP and a single experiment).

For files corresponding to a single MIP and experiment, there are an additional 4 columns specifying the time period needed for each variable and the grid interpolation request. Three columns for the time slice information give the number of years, and type of slice used to specify this and a list of years. The fourth column specifies a grid, of the form "native", "native:01" (a variation on "native" used by OMIP: data should be on the native grid unless it is unstructured, in which case it should be interpolated to regular), "1deg", "2deg". The value that appears here will depend on the "grid policy" (see section 4.7).

The default is to provide variables listed in MIP tables.

Alternatively, variables can be listed by realm and frequency by using the "--xfr" argument.

The variable list include structured information about each variable, including frequency and the CF cell methods attribute specifying any averaging that needs to be done.

Volume Estimate Workings

The volume estimate sheet includes a table with rows for each spatial configuration of data, and a block of 4 columns for each frequency. The 4 columns include a count of variables, the average number of years per variable, and the number of experiments, and finally a volume estimate based on this information. The names of the variables and experiments are included as comments. Sums over rows and columns give the volume associated with each spatial shape and each frequency respectively.

2.2.4 Python Library

```
from dreqPy import dreq
dq = dreq.loadDreq()
```

To load the library in a python session and load the data request into a python object:

The content: dq.coll

The content object, dq.coll is a dictionary whose elements correspond to the data request sections represented as a "named tuple" of 3 elements: "items" (a list of records), "header" (a named tuple – see below) and "attDefn" (a dictionary with record attribute definitions). e.g. dq.coll['var'].items[0] is the first item in the "var" section.

The box shows a piece of sample code to print a list of all the variables defined in the "var" section.

```
from dreqPy import dreq
dq = dreq.loadDreq()
print dq.coll.keys()
print dq.coll['var'].attDefn.keys()
print dq.coll['var'].header.title
print '_'*len( dq.coll['var'].header.title )
print '%20s: %s [%s]' % (
       tuple( [dq.coll['var'].attDefn[a].title for a in
       ['label','title','units']] ) )
for r in  dq.coll['var'].items[:10]:
   print '%20s: %s [%s]' % (r.label,r.title,r.units)
```

The items are instances of a family of classes described below. The "label" etc are available as attributes, e.g. dq.coll['var'].items[0].label is the label of the first record.

dq.coll['var'].attDefn['label'] contains the specification of the "label" attribute from the configuration file. This is also available from the item object itself as, for example, dq.coll['var'].items[0]._a.label.

The following code box shows how this can be used to generate an overview of the content, printing a

```
from dreqPy import dreq
dq = dreq.loadDreq()
for k in sorted( dq.coll.keys() ):
  x = dq.coll[k].items[0]
  for k1 in sorted( x.__dict__.keys() ):
    if k1[0] != '_':
      print '%32s: %s' % (x._a[k1].title, x.__dict__[k1] )
```

sample record from each section, using the “title” of each attribute.

dq.coll['CMORvar'].attDefn['vid'].rClass[9] = 'internalLink': this value indicates that the “vid” attribute of records in the “CMORvar” section is an internalLink and so must match the “uid”[10] attribute of another record. To find that record, see the next section.

The index: dq.inx

The index is designed to provide additional information to facilitate use of the information in the data request.

dq.inx.uid is a simple look-up table: dq.inx.uid[thisId] returns the record corresponding to “thisId”. This is a change from the previous release, in which this dictionary returned a tuple with the name of the section as first element. The name of the section is now available through the “_h” attribute of the record (see next section).

dq.inx.iref_by_uid gives a list of the IDs of objects which link to a given object, these are returned as a tuple of section name and identifier.

dq.inx.iref_by_sect has the same information organised differently: dq.inx.iref_by_sect[thisId].a['CMORvar'] is a list of the IDs of all the elements in 'CMORvar' which link to the given element.

There are also dictionaries for each section indexed by label and, if relevant, CF standard name.

- dq.inx.var['tas'] will list the IDs of records with label='tas';
- dq.inx.var.sn['air_temperature'] give a list of records with standard name 'air_temperature'.

The record object

As noted above, each section contains a list of items. Each item within a section is an instance of the same class. The classes are generated from a common base class (dreqItemBase), but carry attributes specific to each section.

A summary readable summary of a record content can be obtained through the __info__ method. For example, the following commands:

```
>>> i = dq.coll['experiment'].items[0]
>>> i._h.__info__()
```

will yield the following output:

9 Python objects cannot, unfortunately, have attributes with names matching python keywords, so the “class” attribute from the XML document is mapped onto rClass in the pythom API.

10 “uuid” has been replaced with the more general “uid” for “Unique identifier”. Identifiers will still be unique within the document, but will not necessarily follow the uuid specifications.

```
Item <Experiments>: [histALL] __unset__
    nstart: 1
    yps: 171
    starty: 1850.0
    description: * Enlarging ensemble size of the CMIP6 hisorical simulations (2015-2020 under SSP2-4.5 of ScenarioMIP) to at least three members. * DCPP: DCPP proposes a 10 member ensemble of histALL up to 2030 also extended with SSP2-4.5. * Please provide output data up to 2014 as "CMIP6 historical" and 2015-2020 (or 2030 for DCPP) as SSP2-4.5 of ScenarioMIP.
    title: __unset__
    endy: 2020.0
    ensz: 2
    label: histALL
   egid: [exptgroup]Damip1 [a684ca9a-8391-11e5-bca6-0f460b96c0cb]
    tier: 1
   mip: [mip]DAMIP [DAMIP]
    ntot: 342
    mcfg: AOGCM/ESM
    comment:
    uid: a684c950-8391-11e5-bca6-0f460b96c0cb
```

Information about the section and the attributes of records in the section can be obtained through the "_h" and "_a" attributes. For example:

```
>>> i = dq.coll['experiment'].items[0]
>>> i._h.__info__()
Item <X.3 Section Attributes>: [experiment] 1.5 Experiments
    uid: SECTION:experiment
    level: 0
    title: 1.5 Experiments
    id: exp
    useClass: vocab
    maxOccurs: 1
    label: experiment
    itemLabelMode: def
    labUnique: No
sectdef(tag=u'table', label=u'experiment', title=u'Experiments', id=u'cmip.drv.012', itemLabelMode=u'def', level=u'0')
```

Note that in earlier versions the "_h" object was a named tuple, whereas it now has the same basic structure as the record object.

Records to define record attributes (new since 01.beta.11)

Each record contains a collection of attributes with simple names such as "title", "tier": the collection of attributes will be used to define the building blocks of the request. More information about the usage of each attribute is contained in another record which is attached to the parent class. In the above example, for instance, the value of "i.tier" is 1, the specification of the "tier" attribute is in "i.__class__.tier", which is also a record object so that "i.__class__.tier.__info__()" yields the following:

```
Item <Core Attributes>: [tier] Tier of experiment
    uid: __unset__
    title: Tier of experiment
    techNote: None
    label: tier
    superclass: __unset__
    useClass: None
    type: xs:integer
    description: Experiments are assigned a tier by the MIP specifying the tier, tier 1 experiments being the most important.
```

and, because the "tier" object has the same methods as the "i" object, "i.__class__.tier.__class__.type.__info__() " yields:

```
Item <Core Attributes>: [type] Record Type
    uid: __core__:type
    title: Record Type
    techNote:
    label: type
    superclass: rdfs:range
    useClass: __core__
    type: xs:string
    description: The type specifies the XSD value type constraint, e.g. xs:string.
```

This formulation, which embeds all the information, including the definitions of attributes, in the same structure is motivated by the structure of Resource Description Framework (RDF) triples. In RDF and object is defined through a set of triples of the form "object property subject", with the important constraint the "property" must be an RDF object. In the dreqPy implementation the "property" object for "tier" is the record "i.__class__.tier" and the RDF triple is expressed as "i.tier=1".

This feature provides the mechanism for making the API self-documenting. At present there are many attributes which have little or no information in the record "description", but this will be filled out in coming revisions.

The header record for each item is now also an item record with the same structure. The command "i._h.__info__()", where "i" is a record from the "experiment" section as above, yields:

```
Item <Section Attributes>: [experiment] Experiments
    uid: SECTION:experiment
    title: Experiments
    useClass: vocab
    label: experiment
    id: cmip.drv.012
```

Scope.py

Additional module has been added to provide volume estimates. The current draft demonstrates how information can be aggregated, and the basic mechanism for avoiding duplication when multiple MIPs ask for the same data.

The following code will set "x" to the volume, expressed as and estimate of the the number of floating point values, for the C4MIP request with variables up to priority 2:

```
from dreqPy import scope
sc = scope.dreqQuery()
x = sc.volByMip2( 'C4MIP', pmax=2 )
```

The conversion to bytes will depend on the choice of compression, which is not yet represented in the API. The volume for multiple MIPs is obtained passing a python set to volByMip, e.g.

x = sc.volByMip2({'C4MIP', 'LUMIP'}, pmax=2)

An example is provided in “example.py”.

After a call to sc.volByMip2, the variable sc.res['vu'] contains a breakdown of the volume by frequency and the CMOR name of the variable. E.g. sc.indexedVol['mon'].['Omon.thetao'] contains the volume associated with the potential temperature.

The estimate uses a default model configuration. To reset this, change the values in the sc.mcfg dictionary (this part of the module will be improved to support use of a configuration file) with keys given by the labels in Table 1 (section 4.5).

Selection by Tier of Experiments and Priority of Variables

The scope.py module now supports selection of experiments by tier. A call of the following form will configure the “sc” object to consider only experiments with tiers up to, and including, tierMax:

sc.setTierMax(tierMax).

Examples

The “examples” folder in the repository contains a series of example scripts. Documentation for users is provided in “dreqExamples.pdf”[11], which cover both use of the python API and alternative approaches using the XML database directly.

2.2.5 Appendix: list of command line arguments

Data Request Command line.

    -v : print version and exit;
    --unitTest : run some simple tests;
    -m <mip>: MIP of list of MIPs (comma separated; for objective selection see note [1] below);
    -l <options>: List for options:
        o: objectives
        e: experiments
    -q <options>: List information about the schema:
        s: sections
        <section>: attributes for a section
        <section:attribute>: definition of an attribute.
    -h : help: print help text;
    -e <expt>: experiment;
    -t <tier> maxmum tier;
    -p <priority> maximum priority;
    --xls : Create Excel file with requested variables;
    --sf : Print summary of variable count by structure and frequency [default];
    --legacy : Use legacy approach to volume estimation (deprecated);
    --xfr : Output variable lists in sheets organised by frequency and realm instead of by MIP table;
    --SF : Print summary of variable count by structure and frequency for all MIPs;
    --grdpol <native|1deg> : policy for default grid, if MIPs have not expressed a preference;
    --grdforce <native|1deg> : force a specific grid option, independent of individual preferences;
    --ogrdunstr : provide volume estimates for unstructured ocean grid (interpolation requirements of OMIP data are different in this case);
    --allgrd : When a variable is requested on multiple grids, archive all grids requested (default: only the finest resolution);
    --unique : List only variables which are requested uniquely by this MIP, for at least one experiment;

11 proj.badc.rl.ac.uk/svn/exarch/CMIP6dreq/trunk/dreqPy/docs/dreqExamples.pdf

--esm : include ESM experiments (default is to omit esm-hist etc from volume estimates);
--txt : Create text file with requested variables;
--mcfg : Model configuration: 7 integers, comma separated, 'nho','nlo','nha','nla','nlas','nls','nh1'
default: 259200,60,64800,40,20,5,100
--txtOpts : options for content of text file: (v|c)[(+|-)att1[,att2[...]]]
--xlsDir <directory> : Directory in which to place variable listing [xls];
--printLinesMax <n> : Maximum number of lines to be printed (default 20)
--printVars   : If present, a summary of the variables (see --printLinesMax) fitting the selection options will be printed
--intersection : Analyse the intersection of requests rather than union.

NOTES
-----
[1] A set of objectives within a MIP can be specified in the command line. The extended syntax of the "-m" argument is:
-m <mip>[:objective[.obj2[.obj3 ...]]][,<mip2]...]

e.g.
drq -m HighResMIP:Ocean.DiurnalCycle

3. CORDEX Vocabularies

A set of technical documentation for regional climate model diagnostics was developed and made openly available for scientists and technicians in the IS-ENES github project (http://is-enes-data.github.io/). The set includes Data Reference Syntaxes (DRS) and metadata specifications the CORDEX-related activities.

3.1DRS for regional climate modelling in the CORDEX framework.

The first version of this DRS was based on experience from archiving regional climate model (RCM) output in the PRUDENCE, ENSEMBLES and NARCCAP projects. At the early CORDEX stage it was decided that the central CORDEX archive will be a central server and additionally a number of regional data portals, to provide extra specialist support for the regions. When it was decided that CORDEX archiving will be based on ESGF the CORDEX DRS was modified and extended to fulfil ESGF requirements and to synchronise CORDEX archiving with the CMIP5 one as close as possible. However still, since the first version of DRS did not focus on the ESGF archiving there are some inconstancies between CORDEX and CMIP5 metadata and archiving approaches, which may inhibit interoperability of data. The CORDEX DRS (CORDEX Archiving Specifications) specifies technical aspects of CORDEX archive file and data formats, as well as archive content.

See additional information here: http://is-enes-data.github.io/cordex_archive_specifications.pdf

3.2CORDEX variable requirement tables

These tables provide a list of variables requested by CORDEX and describe their metadata (CV). The tables contain a subset of the CMIP5 variables sorted by only output frequencies (daily, monthly, seasonal, 3- and 6-hourly) in contrast to the CMIP5 tables based on frequency and realms. All metadata follows the CMIP5 one and the CF convention.

See additional information here: http://is-enes-data.github.io/CORDEX_variables_requirement_table.pdf

3.3ESGF CORDEX quality control information

This document describes data aspects that have to be checked at data digestion time into an ESGF CORDEX archive. All data aspects to be checked are based on the CORDEX Archive Specifications detailed in 1).

See additional information here: http://is-enes-data.github.io/CORDEX_qc.pdf

3.4DRS for bias-adjusted CORDEX simulations

This document specifies the DRS elements for publishing bias-adjusted CORDEX simulation data through ESGF. It was developed to respond to requests for providing bias-adjusted (or bias-corrected) CORDEX simulations for impact and adaptation studies. This DRS is completely based on the CORDEX one with a number of changes and modifications to include necessary metadata describing bias-adjustment methodology. The DRS for bias-adjusted CORDEX simulation data has the same number of DRS elements in file names as in CORDEX and bias-adjustment information is simply added to one of the CORDEX DRS elements. Such approach allows publishing of the bias-adjusted CORDEX simulation data under the CORDEX-Adjust project on ESGF using the existing CORDEX-ESGF configuration with a minimum of changes. The first bias-adjusted simulations were made available through ESGF in October 2016.

See additional information here: http://is-enes-data.github.io/CORDEX_adjust_drs.pdf

3.5DRS for regional reanalysis

This DRS was mainly developed in the CLIPC project but in very close cooperation with IS-ENES2. The generation of regional reanalyses is very similar to the generation of RCM simulations (nested downscaling in both cases) and the IS-ENES2 experience gained from developing the CORDEX archiving documentation strongly contributed to development of this DRS. Similar to the DRS for bias-adjusted CORDEX data, the DRS for regional reanalysis is based on the CORDEX DRS with changes and modifications necessary to reflect specific aspects of regional reanalysis (e.g. double nesting). The DRS for regional reanalysis can be also used for convection-permitting CORDEX simulations when double nesting is needed.

See additional information here: http://is-enes-data.github.io/drs_reg_reanalysis.pdf

4. Conclusions and Perspectives

The Climate Model Inter-comparison Project, Phase 6, (CMIP6) and the Coordinated Regional Climate Downscaling Experiment (CORDEX) are the leading collaborative projects in global climate research, providing a high profile research resource and underpinning the climate change assessment reports of the Intergovernmental Panel on Climate Change (IPCC). Through the work described in this report, IS-ENES2 has made a major contribution to these activities.

The standards described here will be implemented in data submitted to the CMIP6 and CORDEX archives. The structured approach developed in the CMIP6 data request will facilitate efficient management of complex data standards describing thousands of diagnostics. The CORDEX standards bring a new degree of interoperability to regional data products, enabling more efficient analysis of regional climate projections which are playing an ever increasing role in assessment of climate change and, especially, climate change impacts.