

METAFOR Revised Tools and Services METAFOR Deliverable 5.6 M36

PROJECT	
Project acronym	METAFOR
Project full title	Common <u>Metadata</u> for <u>Climate Modelling</u> Digital Repositories
Grant agreement no:	211753
Funding Scheme	Combination of Collaborative Projects & Coordination and Support Actions
Call Topic	INFRA-2007-1.2.1 Scientific Digital Repositories
DOCUMENT	
Deliverable	D5.6 Month 36
Deliverable Title	Revised Tools and Services
Document Identifier	METAFOR-D5.6 M27
Date	August 30 th 2010
Work Package	WP4 CIM Services
Authors	IPSL
Document Status	Draft
Document Link	http://bit.ly/gS51F

Nature 1		
R	Report	X
P	Prototype	
D	Demonstrator	
O	Other	

Document History			
Version	Date	Comment	Author/Partner
0.1	Apr 20 th 2011	Draft	M. Morgan/IPSL
0.2	Apr 20 th 2011	Draft	A. Treshansky/UMKO

Table of Contents

1. SUMMARY.....	3
2. WEB SERVICES.....	4
OVERVIEW.....	4
SERVICE OPERATIONS.....	4
3. WEB APPLICATIONS.....	5
CMIP5 QUESTIONNAIRE.....	5
CMIP5 QC TOOL.....	5
CIM WEB PORTAL.....	5
CIM GEONETWORK.....	6
4. COMMAND LINE UTILITIES.....	7
CMIP5QINGESTOR.....	7
CMIP5QINGESTOR.....	7
CIMGEONETWORKINGESTOR.....	7
TDS2CIM.....	7
CIMFROMCODE.....	7
5. ROADMAP.....	8
30/04/2011.....	8
31/08/2011.....	8
30/09/2011.....	8

1. Summary

The METAFOR Common Information Model, i.e. the CIM, is an ontology designed to become the ipso-facto standard for climate modelling related metadata. METAFOR is responsible for incubating a CIM ecosystem, i.e. a set of useful CIM compliant tools & services. Such an eco-system is essential to encouraging adoption of the CIM by the global climate modelling community

CIM compliant tools & services consist of a collection of software artefacts designed to standardise the operational activities of climate modelling institutes. Such activities include model documentation, output data discovery, quality control, experimental design, code documentation, coupling configuration, etc. The software artefacts have been developed in the Python programming language and can be divided into 3 categories:

1. Web services;
2. Web applications;
3. Command line utilities.

Such tools & services are necessarily inter-dependent as in the large part they either share data or common components. For example, the CIM search web service executes queries against a back-end database populated with CIM compliant XML instances ingested from the CMIP5 Questionnaire web application. Another example is the THREDDS to CIM command line utility that publishes it's output to the CIM instance web service. Thus whist each individual tool/service delivers a functional subset, collectively the tools & services deliver a platform, at the heart of which is the CIM. This document will pay particular attention to the CIM Portal web application which integrates together several web services and provides a focal point for users wishing to find and interact with CIM instances.

METAFOR tools & services in support of the CIM continue to be under active development as of April 2011. Active development will continue during the METAFOR project extension phase at which point a transition of the work to the IS-ENES project will occur (handover planning for this has already begun). A roadmap outlining the development plan for the remaining months of the METAFOR project is presented along with details of the various tools & services either delivered or planned to be delivered.

2. Web Services

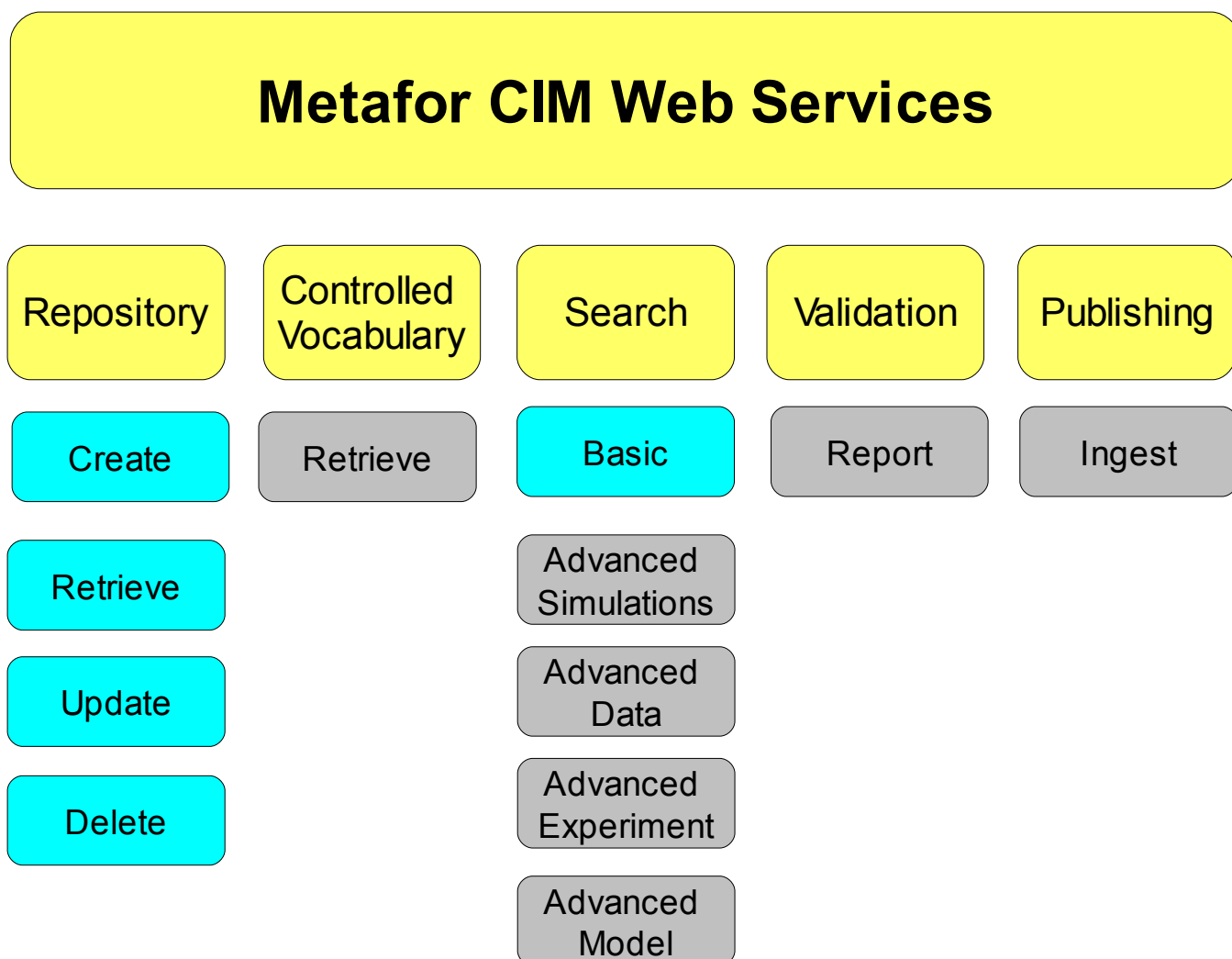
Overview

The CIM web services have been implemented according to the Service Orientated Architecture (SOA) paradigm. SOA is a key tenet in the design of secure, robust and distributed systems. Such an architecture deconstructs a system into a set of discrete functional units known as web services. Such services are supplied by providers (i.e. METAFOR) and consumed by clients (i.e. portals, tools, applications, and in some cases other web-services).

The CIM web services are designed so that institutes can gradually over time integrate them into their own climate modelling workflows. For example an institute may wish to automatically publish meta-data to the CIM web services so that this meta-data will subsequently appear in the CIM Portal (see below).

Service Operations

Below are outlined the set of CIM web service operations. Operations in cyan have already been implemented. Operations in grey will be implemented during the project extension period. For further details see METAFOR EU deliverable 5.5 *CIM Web Service Implementation* [http://bit.ly/metafor_D55].



CIM Repository Implementation

Most of the CIM web service operations interact with a back-end eXist XML database which can be queried and/or manipulated using XQuery functions, XQuery being an XML query language. The CIM web services invoke these XQuery functions whenever it needs to interact with the database.

The available XQuery functions include: retrieving a document; removing a document; creating/updating a document either by passing the document as a string or a URL (that resolves to a CIM document) or a file

object; re-indexing the database; performing a “basic” search; basic differencing. Functions in development include “advanced” search and extending the differencing capabilities.

Each CIM document has a id plus version that uniquely identifies the CIM document within the database, and are what gets passed as parameters to the retrieve, remove, and create/update functions. Any user can retrieve a CIM document. Indeed, retrieving documents is a required step of many of the existing web-services. Similarly, creating/updating CIM documents forms part of the CIM ingestion processes described elsewhere in this document. However, the ability to remove a CIM document should be restricted to users with the appropriate privileges.

The remaining XQuery functions can be configured by using “facets.” A facet is simply a named Xpath expression pointing to some part(s) of a CIM document. Facets are used in four ways by the back-end:

1. To define the set of keywords to use for the “search suggest” functionality in the front-end.
2. To define the summary information to return for CIM documents that match a query.
3. To define the subset of CIM information to query for different types of “advanced” search.
4. To define the subset of CIM information to compare during differencing.

These facets are stored in a set of XML configuration files that are themselves stored in the eXist database. The precise details of what the XQuery functions return can be modified by adding, deleting, or changing the facets in those files. This is much more straightforward than rewriting XQuery code – it requires only a basic knowledge of XPath and does not require restarting or otherwise reconfiguring the back-end.

“Basic” search lets users select a document type (model, simulation, dataset, etc.) and a *single* facet and issue a query to the database. “Advanced” search will let users search multiple facets at once; For example, searching for all documents which include a grid type of “arakawa-c” and a name of “hadgem2-es” and an output file containing “SST.” Note that each facet is associated with a separate text query. The front-end will provide a webform for users to craft an advanced query. The particular facets to use for an advanced query will be defined in the aforementioned configuration files. In contrast, a single combo-box is all that is needed for users to select the single facet in basic search. Currently, neither the front-end for advanced search nor the set of facets to use for advanced search have been completed.

In both basic and advanced search, results will be returned as an XML document providing summary information of each matching document by the back-end, and rendered as an HTML table (with added JavaScript functionality) by the client, e.g. the CIM portal. Actually, the results are displayed as a series of tables, each one corresponding to another CIM document type. Every row of each table corresponds to another matching document. The columns of the tables correspond to the results of retrieving the values of the facets (for summaries of *that* type of document). For example, if a facet exists for model components called “child components” with an XPath expression giving the name of all child components in a CIM document, then in the result table for model components a column will exist called “child components” with the content of each cell in that column being the result of evaluating that XPath expression for the document corresponding to that row. Of course, raw XML will not be displayed in the table. Instead, the facet provides information to the front-end on how it should be displayed. In this case, the facet would indicate that it is a tree of strings and one would expect the front-end to use a treeview widget or something similar to render it. Any table row can be selected to view the corresponding CIM document in greater detail. This will launch the CIM Viewer service mentioned above. And any pair of those documents can be selected to run a comparison against. This will launch the CIM Difference service.

Differencing a pair of CIM documents can mean many things. At one extreme, it could be a simple text-based differencing of two XML files. However, that is unlikely to provide users with much useful information. Instead, facets are used to define particular subsets of CIM instances (of particular document types) that should be compared; The rest of the instances can be ignored during differencing. Each differencing facet can also specify a comparison function to use. By default, all facets check to see if elements and attributes are equal. However, one can imagine situations where it would be more useful to check if dates or numbers are in a particular range when determining whether or not differences exist. The XQuery differencing function returns an XML report describing each facet that was examined, the value of evaluating that facet for each document being compared, and if there was an “addition” or “deletion” or “modification” (as defined by a specific comparison function) at that facet. Differencing is a recursive operation; If a facet evaluates to a node with children, then those nodes are all checked in turn. The front-end for rendering a differencing report is still being developed.

The eXist database incorporates Apache's Lucene search-engine library which makes running the queries particularly efficient and also provides built-in support for wild card and regular expression searching. However, it requires effective indexing of the database. In particular, where there are facets which define nodes within a CIM document to search, those nodes need to be added to the index. An XQuery reindexing function does this automatically; it parses the configuration files and generates all appropriate indices. It also checks the contents of the database to see if those nodes use particular namespaces. If so, those namespaces are incorporated into the new index. There is no Python web-service provided for the back-end reindexing function. It is therefore not publicly accessible. It should be called as needed by a CIM Portal administrator when facets are modified or when documents with new namespaces are ingested into the database (as when new versions of the CIM are released).

3. Web Applications

CMIP5 Questionnaire

The CMIP5 Questionnaire is in production and is being used by climate modelling institutes worldwide. It collects meta-data from institutes participating in the CMIP5 process. Users are guided through a series of webforms and asked to enter information describing their climate models, simulations, and data. Entered content is being ingested by both the ESG Curator portal (see <http://www.earthsystemcurator.org/>) and the CIM Web Portal (see below) as CIM instances. Available instances can be published as ATOM feeds. In the case of the CIM Portal, once a feed has been registered it is periodically inspected to see if there is new content that should be ingested into its back-end database. The CMIP5 Questionnaire can be found online at <http://q.cmip5.ceda.ac.uk/>. Development is ongoing.

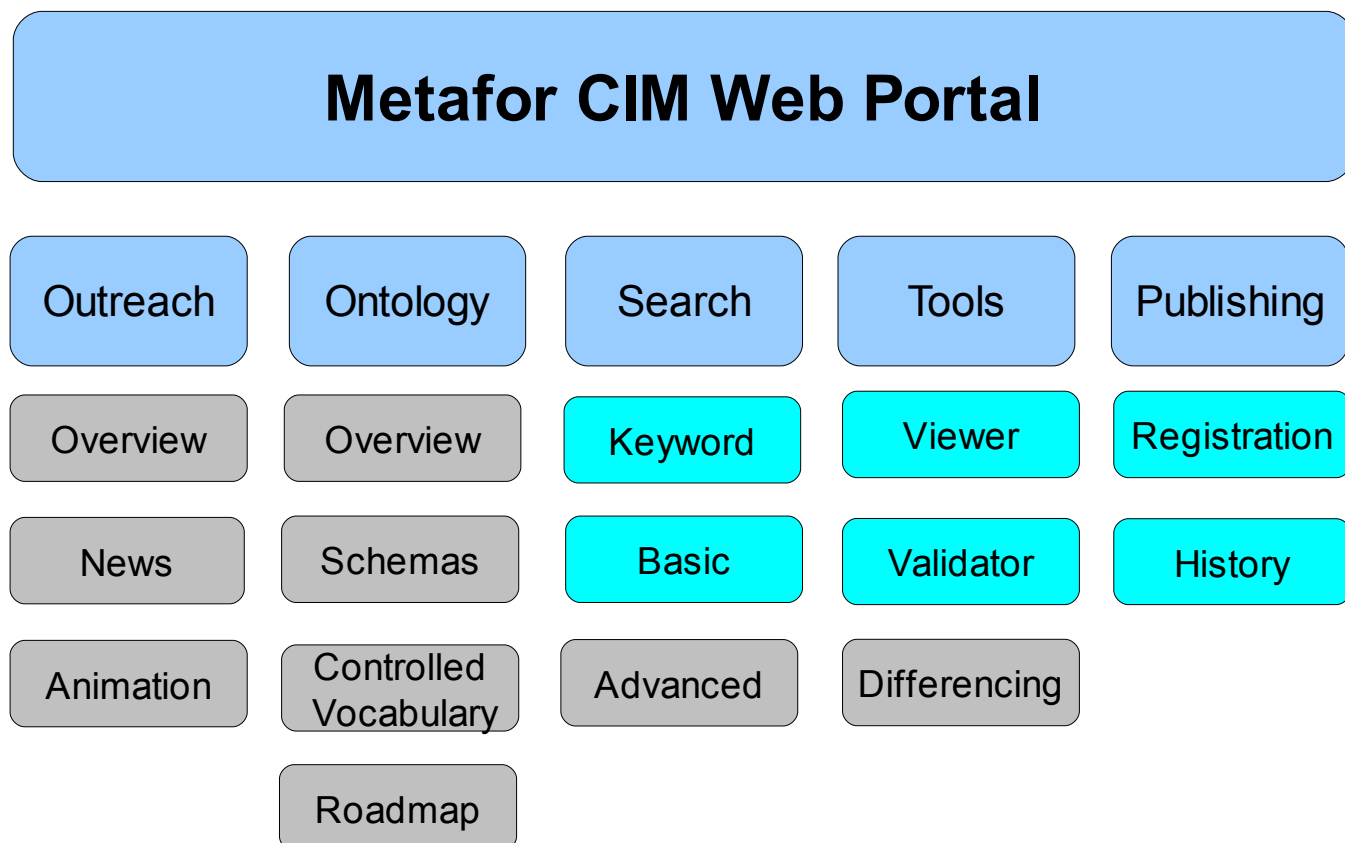
CMIP5 QC Tool

The CMIP5 QC (Quality Control) Tool is in production and has been developed under the auspices of METAFOR. It assists the community to document quality related issues connected with CMIP5 output. It exposes it's content, i.e CMIP5 output quality control issues/reports, as CIM compliant XML documents embedded within an Atom feed. It can be found online at <http://qc.cmip5.ceda.ac.uk/qc>. For further information please refer to its wiki page [http://bit.ly/cmip5_qc_tool].

CIM Web Portal

The CIM Portal is designed to deliver a diverse set of use cases to a diverse set of actors. The set of supported use cases can be placed into several categories: outreach, ontology, search, tools and publishing. The set of actors can be placed upon a spectrum of familiarity with CIM meta-data ranging from members of the public to climate or metadata experts.

Below is outlined the CIM Portal site plan. Site functions in cyan have already been implemented. Site functions in grey will be implemented during the project extension period. For further details see METAFOR EU deliverable 5.5 CIM Web Portal Deployed [http://bit.ly/metafor_d43].



CIM Geonetwork

GeoNetwork provides a GUI for the editing of XML documents. It is an Ajax based standardized and decentralized spatial information management environment. Any kind of data that is based on an XML schema can be made available inside the GeoNetwork framework. This XSD file describes everything about the structure of a record. The GeoNetwork software analyses the XSD schema during start up so that the appropriate structures and settings appear when a record is displayed in 'Show' or 'Edit' mode e.g. an enumeration list in the schema is shown as a selection list at the screen.

A GeoNetwork server in support of the CIM has been set up by the DKRZ group in Hamburg. CIM instances hosted by this server are eligible for ingestion into the CIM Web Portal (see above) and thus are discoverable.

4. Command Line Utilities

Cmip5QIngestor

Ingests CIM compliant information exposed by the CIMP5 Questionnaire atom feeds. Ingested information can be persisted to a back end XML database for use within the CIM Web Portal.

Cmip5QCIngestor

Ingests CIM compliant information exposed by the CIMP5 QC Tool atom feeds. Ingested information can be persisted to a back end XML database for use within the CIM Web Portal.

CIMGeonetworkIngestor

Ingests CIM compliant information exposed by the CIM GeoNetwork server. Ingested information can be persisted to a back end XML database for use within the CIM Web Portal.

TDS2CIM

Parses a THREDDS metadata server and converts discovered metadata into CIM DataObject documents. Can optionally publish the generated documents to CIM web services. This utility can potentially play a key role in aggregating output data. Development is ongoing.

CIMfromCODE

Under development by a group from the University of Manchester. This utility aims to support model code (i.e. Fortran) documentation by parsing Fortran files, extracting metadata, converting the extracted metadata to a CIM compliant structure, and optionally publishing via CIM Web Services (see above) to the CIM Web Portal (see above). Development is ongoing.

5. Roadmap

30/04/2011

Web Services	CIM Web Services	In Progress
Web Applications	CMIP5 Questionnaire	Complete
	CIM QC Tool	Complete
	CIM Geonetwork	Complete
	CIM Web Portal	In Progress
	Command Line Utilities	Cmip5QIngestor
Deployments	Cmip5QIngestor	Complete
	CIMGeonetworkIngestor	Complete
	TDS2CIM	In Progress
	CIMfromCODE	In Progress
	CMIP5 Questionnaire	Production
	CIM QC Tool	Production
	CIM Geonetwork	Test
	CIM Web Services	Test
	CIM Web Portal	Test

31/08/2011

Web Services	CIM Web Services	Complete
Web Applications	CMIP5 Questionnaire	Complete
	CIM QC Tool	Complete
	CIM Geonetwork	Complete
	CIM Web Portal	Complete
	Command Line Utilities	Cmip5QIngestor
Deployments	Cmip5QIngestor	Complete
	CIMGeonetworkIngestor	Complete
	TDS2CIM	Complete
	CIMfromCODE	Complete
	CMIP5 Questionnaire	Production
	CIM QC Tool	Production
	CIM Geonetwork	Production
	CIM Web Services	Production
	CIM Web Portal	Production

30/09/2011

IS-ENES Handover