CIM-enabled OASIS

METAFOR Deliverable 5.7 M36

PROJECT	
Project acronym	METAFOR
Project full title	Common Metadata for Climate Modelling 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	D5.7 Month 36
Deliverable Title	CIM-enabled OASIS
Document Identifier	METAFOR-D5.7_M36
Date	April 12th, 2011
Work Package	WP5 Development of a toolkit to manipulate the CIM
Authors	CERFACS
Document Status	Final
Document Link	http://metaforclimate.eu/documents

Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants	
RE	Restricted to a group specified by the Consortium	
CO	Confidential	

Abstract

This report describes the adaption of the OASIS4 coupler so that it can use a CIM XML file for its configuration instead of its original XML files. For each element of the original XML configuration files, the equivalent element or attribute in the CIM was identified or the CIM was complemented. Then the sources of the OASIS4 coupler were modified so that it could read a CIM XML file. The work was then validated by running two toy models using a CIM XML file as the configuration file.

1. Introduction

The Common Information Model (CIM) is a formal metadata model of the climate modelling process. It includes descriptions of the experiments being undertaken, the simulations being run in support of these experiments, the software models used to run these simulations and the data produced as well as required by these models. In particular for OASIS, the CIM can be used to provide *coupling metadata.*

The OASIS coupler is a software tool which allows coupling fields to be exchanged in a synchronized way between component models which can remain independent executables. It is

able to perform transformations and interpolations between component grids in order to map fields from a source to a target component when those components' grids are different. In order to do this, a formal metadata description of fields to be coupled, the components those fields belong to, the structure of those components' grids, the timing of coupling exchanges within the simulation, and the transformations to be performed by the coupler on the fields are required.

In general, it is important to be able to precisely describe the coupling exchanges between the components of an Earth System model (ESM) because these exchanges determine how the components interact and therefore define a part of the physics modeled by the ESM. In particular, it is important to describe for each coupling exchange the source and the target components, the coupling frequency, the transformations performed on the data produced by the source component on its mesh to express it on the mesh of the target component, etc. This description of the coupling exchanges should be generic and independent of how it is implemented, i.e. through separate coupling software such as the OASIS4 coupler, via argument passing, or via shared modules.

The CIM describes coupling by associating a “composition” with a top-level software component. A composition describes a set of couplings. Each coupling maps a data source to a data target, where a source or target can either be another component or a data file. A coupling also describes a set of connections. Each connection also maps a data source to a data target; in the case of a connection, though, the source and data can either be a component property (ie: a field or variable modeled by a component) or the content of a data object (ie: a field or variable stored in a data file). Thus, a coupling defines an input/output channel between high-level components (or files) and a connection defines a sub-channel between the low-level contents of those components (or files). Both a coupling and a connection can have timing information associated with them describing how often an exchange takes place. Also, both a coupling and a connection can have transformation information associated with them describing any modifications to the exchange fields that will be performed to map from the source to the target grid. These transformations can either be described as a separate component (i.e. a connection exists between atmos and the atmos-to-ocean transformer *and* between the atmos-to-ocean transformer and the ocean), or by various “inline” attributes that are part of the coupling or connection itself. If a coupling is “fully-specified,” then all of the exchanges between the source and target component are described as connections. For many user groups, this level of detail is unnecessary. For OASIS, it is required.

When originally developing the OASIS4 coupler, it was decided that the person setting up a coupled system using this coupler would have to describe the coupling exchanges in few XML files. A “Specific Coupling Configuration” (SCC) file contains the general characteristics of a coupled model run, e.g. the start and end date of the run and the components to be coupled; then for each component model a Specific Model Input and Output Configuration (SMIOC) specifies the relations the component model will establish with the external environment through inputs and outputs.

The objective of the current deliverable was to adapt the OASIS4 coupler so that it could use a CIM XML file for its configuration instead of the SCC and SMIOCs XML files. The first step was to identify for each element or attribute of the SCC and SMIOC file the equivalent element or attribute in the CIM or to complement the CIM when it was missing; this step is described in section 2 detailing changes in the semantics and in the metadata structure. The second step, presented in section 3, was to modify the sources of the OASIS4 coupler so that it could read a CIM XML file instead of the SCC and SMIOC files. The third step was to validate the work done by running two toy models using a CIM XML file as the configuration file; this last step of course implied creating the corresponding CIM XML file based on the SCC and SMIOC files existing for these toy models, as described in section 4. The rest of the document describes into

more details the work done to successfully reach the original objective and concludes in section 5 with a discussion on the CIM benefits and costs for OASIS4 and on future perspectives.

2. Mapping between the CIM and OASIS4 original configuration files

At first, an exact mapping of OASIS4 configuration data with the CIM version 1.4 did not exist. However, there were correspondences for the main features. In moving from CIM v1.4 to v1.5, many new features were added specifically to support OASIS4 and the OASIS4 managed to find a place for all OASIS configuration metadata in the CIM. Appendix A presents the detailed mapping between all SCC and SMIOC configuration elements and attributes and the CIM elements and attributes. Some particular changes in the semantics and in the metadata structure introduced for the mappings are discussed here in more detail.

Extension of the CIM

In few cases, for information that was not OASIS4-specific, new metadata structures were added to the CIM. In Appendix A, they can be spotted by "OK-ADDED". One example is the number of processes used to run the OASIS4 coupler and the different components which now appear as the element <processes> in the <parallelisation> element of a <deployment>.

When OASIS4 needs were too specific, multi-purpose metadata structures were added to the CIM. For example, the specific spatial regridding methods and options applied on the coupling fields can now be defined in a multi-purpose <spatialRegriddingProperty> within <spatialRegridding>.

Reduction of the OASIS4 configuration metadata

Some non mandatory options, previously set by the OASIS user, were completely removed from the OASIS4 configuration, as for example the option to gather or dispatch debug trace information in one or several files; a default value is now used for these options. Other information that can be deduced or calculated by OASIS4 has simply been stripped out of the OASIS4 configuration; this is the case, for example, for the periodicity of a grid (attribute periodic in element <indexing_dimension>.

Changes in the metadata structure in the XML files

In the original configuration, the metadata information is distributed over several files: one SCC and several SMIOC, one per component model. With the CIM, the whole information can be gathered into only one CIM document. This one document is still organised internally by component model, but many difference still exist at lower levels.

Some of those are of small importance. For example, the information may be encoded in an XML element instead of an XML attribute, or the information needs to be defined at a deeper level in the XML element hierarchy, or the words used in the CIM differed from those found in the SCC or SMIOC (for example, “TimeAverage” instead of “taverage”).

But many pieces of information in the CIM, due to their standardisation across several communities, have a more complex structure than in the SCC or SMIOC. A CIM data element has to cover all possible cases of existing coupling model. The CIM data structure that describes such element is therefore more complex and larger. For example, according to the CIM, NetCDF files must appear in a <DataObject> element that has been completely standardised and has a rather complex structure.

3. OASIS4 source modification to read CIM XML files

A complete list of file changes may be found in Annex B at the end of this document. The modified sources were checked in the OASIS4 SVN repository and are available at:
https://oasistrac.cerfacs.fr/browser/branches/development_4/oasis4

The goal was to modify OASIS4 sources to support both SCC/SMIOC and CIM configurations. Care has been taken not to disrupt current software functionality. By default, the SCC/SMIOC configuration is used. If the user wants to use the CIM configuration, he has to compile the OASIS4 source code with the compiler flag -DCIM.

Both the CIM and the original SCC/SMIOC configuration files are written in XML. Because of this, the low-level code in OASIS responsible for reading XML documents (sasa_c_xml.c and sasa_c_f90.c) could be kept as is. Also, the main data structures used to store the configuration metadata in OASIS4 memory were not changed. For example, although connections are defined in the CIM as whole objects, OASIS stores the related information in memory in two parts, one for each end of the connection. The sequence of actions performed by OASIS to serialise this configuration data was kept the same, i.e. first the reading of general coupling metadata, then the reading coupling metadata specific to each model component. That last step is still divided into two phases, one for the connection source parameters and one for the connection target parameters.

In the original OASIS4 software, two modules are related to loading configuration data into memory, one for reading SCC file (*psmile_scc.F90)* and one for reading SMIOC files (it is composed of several files whose names begin with get_ as for example *get_smioc_numbers.F90, get_transi_details.F90, get_grids_details.F90, ...*)

In CIM related software, *psmile_CIM.F90* replaces *psmile_scc.F90* for reading general coupling configuration. For reading specific coupling configuration, we kept the *get_xxx* files but their content has been extended with Fortran code related to the CIM.

We also extended the number of get_xxx files:

- get_regrid_details.F90 is devoted to reading parameters for regridding field data
- get_cnct_source_details.F90 is responsible for reading parameters of a connection at the source point
- get_cnct_target_details.F90 is responsible for reading parameters of a connection at the target point

We also removed some miscellaneous source files, mainly for simplification. Their source code, modified in the CIM version, has been simply moved into some other existing source files:

- prismdrv_get_smioc_file_name.F90
- get_smioc_grids_transi_nb.F90
- get_smioc_transi_numbers.F90.

4. Validation of the CIM configuration

The modifications brought to OASIS4 sources to use the CIM as configuration file were validated with two existing toy coupled models, toyoa4 and tutorial1.

The toyoa4 scenario couples three toy component models and reproduces the exchanges between land, ocean and atmosphere components, using the grids of real components. Since the atmosphere and ocean do not use the same grid, OASIS4 performs some regridding on the fields exchanged between these components. This toy model also implements time transformation on some exchanged fields. It is a rather complete test case for OASIS4.

The original SCC and SMIOC configuration XML files were first converted to the CIM format. The resulting CIM file is available at:
https://oasistrac.cerfacs.fr/browser/branches/development_4/oasis4/examples/toyoa4/input/cim.xml.start

The toy model toyoa4 was run and the results were compared with the toy model configured in the standard way. Toyoa4 includes some I/O to and from NetCDF files; a detailed comparison between standard and CIM version of the toy coupled model output NetCDF files showed that their content is identical. OASIS4 also offers an option for outputing field data before or after exchange. This option was turned on for several exchanged fields and the resulting output files were compared: their content was identical in both the standard and

CIM versions. Finally, OASIS4 offers an option for writing out statistics on field data before or after exchange. We turned on this option for several exchanged fields and compared their statistics: they are identical in both standard and CIM version.

Tutorial1 is another example provided with OASIS4. It implements a basic test exchange between two components. The data exchanged may be a single field or a bundle of fields. We used the bundle version, converted the configuration data into CIM format and ran the example. The resulting CIM file is available at

https://oasistrac.cerfacs.fr/browser/branches/development_4/oasis4/examples/tutorial1/data_oasis4/cim.xml_bundle

After comparison, as with the toyoa4 example, the results were identical.

5. Conclusion and discussion

The original objective to adapt the OASIS4 coupler so that it could use a CIM XML file for its configuration instead of the original SCC and SMIOCs XML files were reached and the modifications were validated with two test examples. Running these examples, many features of OASIS4, like regridding, time transformation, bundling of fields and I/O, and debugging output, were tested.

The one disadvantage of using a CIM XML file to configure OASIS4 coupling exchanges is that the structure of the CIM is more complex and flexible than the SCC and SMIOC. This means that quite often there is more than one way in the CIM to describe the same thing. Therefore, the OASIS4 sources reading the CIM XML files are more complex as they have to cover all cases. For example, the different components implied in a coupling could in principle be described as separate CIM documents and a composition could reference the external documents. As a first step, we limited ourselves to the cases where all components are described in only one CIM xml document.

The main benefit of using a CIM XML file, however, is the use of a standard format. In the long term, should the CIM becomes a widely adopted format like METAFOR aims, coupled model developers and users will not have to become familiar with a configuration format specific to OASIS4 but will simply be able to use the standard metadata format used within climate modeling.

Another advantage is that the definition of the coupling exchanges (called *connections* in the CIM) between any two components is simpler in a CIM compliant configuration file than with SMIOC files. As described above, each component has its own SMIOC file and a user must describe the connection in both the source and the target components' SMIOCs. This redundancy is a potential source of incoherency. Using the CIM, a user defines a connection as only one specific object that makes reference to both connection ends.

Another obvious advantage is the possible “factorisation” of the connection attributes. Several connections between one source and one target component can be grouped into a single coupling. If the connections between this source and this target component share some properties, then those properties may be defined just once at the coupling level.

Technically speaking, using the CIM metadata format for OASIS also has advantages for the SCC and SMIOC format. The lessons learned in developing the CIM can inform these earlier attempts at defining general coupling metadata.

Appendix A : OASIS4 to CIM mapping details

Keys to read the document:

- OK = OASIS to CIM mapping found in the CIM. No adaptation required.
- OK-ADDED = the CIM has been adapted to ensure the mapping (adding a new CIM element).
- OK-ALLOWED = the CIM has been adapted to ensure the mapping (cardinality of an existing CIM element has been changed).
- OK-MOVED = the mapping is addressed but CIM corresponding element has been displaced in the meantime.
- OK-OASIS = the mapping is no longer necessary (eg. element suppressed in OASIS).
- TODO-OASIS: there is no direct mapping. OASIS has to build the information he needs from other existing CIM elements.
- @"name" is for simple CIM elements or attributes (not complex, expecting only one value)
- <name> is for complex CIM elements (containing other XML elements or attributes)

OASIS4 parameter	**CIM**
SCC	
<experiment>	OK: will be specified as a **<simulationComposite>**
<driver>	OK : will be specified as a **<processorComponent>**
<nbr_procs>	OK-ADDED: @**processes** has been added in the **<deployment>/<parallelisation> in <processorComponent>**
<start_date>	OK: @**startPoint in <simulationComposite>**
<end_date>	OK: @**endPoint in <simulationComposite>**
@local_name	OK: @**shortName in <simulationComposite>**
@long_name	OK: @**longName of <simulationComposite>**
@start_mode *Possible values: "spawn", "not_spawn"*	OK: **<componentProperty> of the <processorComponent>** « driver » (@shortName = start_mode)
<run>	OK: will be specified as a **<simulationRun>**
<start_date>	OK: @**startPoint in <simulationRun>**
<end_date>	OK: @**endPoint in <simulationRun>**
<application>	OK: will be specified as a **<modelComponent>** in a **<childComponent>** in a **<modelComponent>** in a **<model>** (which represents the whole coupled model) in **<SimulationRun>** in **<child>** in **<SimulationComposite>**
<argument>	OK-ADDED: @**executableArgument in the <deployment>** element

OASIS4 parameter	CIM
<host>	OK-MOVED: element **<platform>** i**n <deployment>** (<unit> finally not used)
<nbr_procs>	OK-ADDED: @**processes** has been added in the **<deployment>/<parallelisation>**
<local_name>	OK-MOVED: @**shortName** in **<deployment>/<platform>**
An application can be run on different <host>, each running in parallel several processes (<nbr_procs>)	OK-ALLOWED: **multiple <deployment>** elements each one with a <parallelisation> element *Explanation: an application parallelised using MPI may be run on an heterogeneous network. Each running computer host may be a different platform. However, OASIS4 needs to specify the name of each running host.*
<component> <rank> <min_value> <max_value> <increment>	OK-ADDED: @**rankMin,** @**rankMax** and @**rankIncrement** added in <**deployment>/<parallelisation>/<rank>** along with @rankValue
@local_name	OK: @**shortName of a <componentProperty>** of the <modelComponent> “application”
@executable_name	OK-ADDED: possibly several @**executableName into the <deployment>** element
@redirect	TODO-OASIS: **Set to “true” default value** in OASIS4.
SMIOC	
<component>	OK: **<modelComponent>** in **<childComponent>** in a **<modelComponent>** (which represents the application) in **<childComponent>** in a **<modelComponent>** in a **<model>** (which represents the whole coupled model) in **<SimulationRun>** in **<child>** in **<SimulationComposite>**
@local_name	OK: @**shortName of the <modelComponent>** “prismcomponent”
@long_name	OK: @**longName of the <modelComponent>** “prismcomponent”

OASIS4 parameter	CIM
<code> <laboratory> <contact> <documentation>	OK-OASIS: not used.
<grid> @local_name <indexing_dimension> @index @periodic	OK-OASIS
<transient>	**OK:** will be specified as a **<componentProperty>**
@local_name	**OK:** @**shortName of the <componentProperty>** “transient”
@long_name	OK: @**longName of the componentProperty** « transient »
<transient_standard_name>	OK: @**standarName of the <componentProperty>** “transient”
<intent> **<input>**	OK: @**intent in the <componentProperty>** “transient”
<exchange_date>	OK: **<timeProfile>/<rate> of a <coupling>** OR **a <connection>.**
<origin> <corresp_transi_out_name>	OK: @**name of <reference> in <connectionSource>/<dataSource>** *<u>Note:</u> Without using a reference (more complex) : @shortName of <connectionSource>/<dataSource>/<softwareComponent>*
<file> OR <component_name>	OK: @**name of <reference> in <couplingSource>/<dataSource>** *<u>Note:</u> Without using a reference (more complex) : @fileName of <couplingSource>/<dataSource>/<dataObject>. /<storage>/<fileStorage>* OK: @**name of <reference> in <couplingSource>/<dataSource>** *<u>Note:</u> Without using a reference (more complex) : @shortName of <couplingSource>/<dataSource>/<softwareComponent>*

OASIS4 parameter	CIM
<middle_transformation> <interpolation>	OK: **<coupling>/<spatialRegridding>** OR **<connection>/<spatialRegridding>**
<interp3D> OR <interp2D> + <interp1D>	OK: **@spatialRegriddingDimension in the <spatialRegridding> of the <connection>** OR **of the <coupling>** OK-ADDED: the ability **to have 1...3 (or more) <SpatialRegridding> (with each their own @spatialRegriddingDimension)** OK-ADDED: "**1D"** to the enumeration list for **@spatialRegriddingDimension** (curently "2D", "3D") *So we could describe a regridding as :* *3D* *2D (horizontal) + 1D (vertical)* *1D + 1D + 1D*
<interp3D> *Possible choices: <nneighbour3D> <trilinear>* <interp2D> *Possible choices: <nneighbour2D>, <bilinear>, <bicubic>, <conservative2D>* <interp1D> *Possible choices: <linear>*	OK-ADDED/RENAMED: **<spatialRegriddingStandardMethod> added to <spatialRegridding>** and whose value list could be : 4. linear 5. near-neighbour 6. cubic 7. conservative-1^{s}order 8. conservative-2nd-order OK:: @conservativeSpatialRegridding attribute and @spatialRegriddingOrder
Each of the above choices may have further properties: <para_search>, <if_masked>, <order>, <normalisation2D>, <bicubic_method>, <gaussian_variance>, <methodnorm2D>	OK-ADDED: a multi-purposed **<spatialRegriddingProperty>** with name&value **in <spatialRegridding>** (similar to <connectionProperty> request below)

OASIS4 parameter	CIM
<interp3D> <user3D> <name> <format> <io_mode> *Possible values for <io_mode>: iosingle, parallel, distributed*	OK-ADDED: **<spatialRegriddingUserMethod> added to <spatialRegridding>** and containing a **<file>** element **of type dataObject.** OK: **storage>/<fileStorage>/@filename of <dataObject> <distributionFormat>** of **<distribution>** of **<DataObject>** OK: **dataProperty of the <dataObject>** with name&value (ex. @name=io_mode, @value=iosingle)
@transi_in_name	TODO-OASIS: will be **reconstructed based on the @shortName of the <componentProperty>** "transient" and the number of connections this transient is involved in as a <connectionSource>.

OASIS4 parameter	CIM
<target_transformation> **<target_local_transformation>** <gathering> <add_scalar> <mult_scalar> <u>Note:</u> *OASIS may automatically recognize <gathering> as a transformation to be performed on the target but not the other ones (<add_scalar> and <mult_scalar>)*	OK-ADDED: create a multi-purposed **<connectionProperty>** sub-element of <connection> element Such a <connectionProperty> would have only two sub elements : @name and @value. Same thing for <coupling>: add **<couplingProperty>** element *TO-ADD: we realized that having a [connection/coupling]Property is not sufficient. We must distinguish the transformation operating on the source component from the one operating on the target. In order to avoid prefixing the transformation names by "target_" or "source_" we ask for adding <connectionSourceProperty> to the <source> element of a connection and <connectionTargetProperty> to the <target> element of a connection.* *Same thing for <coupling>: <couplingSourceProperty> and <couplingTargetProperty>* *Note : Allyn would prefer we use a <transformer> (of type ProcessorComponent> for describing regridding and time transformations of couplings/connections.* OK-ADDED: Allyn finally agrees with our R26/27 request and implement it by adding a <property> of type **[coupling/connection]Property** to **Endpoint** type. EndPoint being now the type for **<couplingSource>** and **<connectionSource>** (=BFG request to introduce an @instanceID to distinguish different instances of components involved in a coupling). So the mapping is: **<connection>/<connectionSource>/<connectionProperty>** with name&value **OR** **<coupling>/<couplingSource>/<couplingProperty>**
<target_time_operation> *Possible values:* *« time_nneighbour »,* *« time_linear »*	OK: **<timeTransformation> of a <connection> or <coupling>** TODO-OASIS: The values for time operations are distinct according whether they occur on the source or on the target. OASIS will perform an automatic analysis of the value and **assign the time_operation to he right place (source our target)** REMOVE: "lagged" and "none"

OASIS4 parameter	CIM
<statistics> <masked_points> <notmasked_points> <all_points>	OK-ADDED: each will be expressed as a **<coupling>/<couplingTarget>/<couplingProperty>** with name&value **OR a <connection>/<connectionTarget>/<connectionProperty>** (cf. **EndPoint** type)
<debug_mode>	OK: will be expressed as a **<connectionTarget>/<connectionProperty>** with name&value OR **<couplingTarget>/<couplingProperty>** (cf EndPoint type)
<output> <exchange_date> <corresp_transi_in_name> <file> OR <component_name>	OK: **<timeProfile>/<rate> of a <coupling> or a <connection>.** OK: **@name of <reference> in <connectionTarget>/<dataTarget>** OK: **@name of <reference> in <couplingTarget>/<dataTarget>** OK: **@name of <reference> in <couplingTarget>/<dataTarget>** *Note: cf. <input> for a more complex solution without using <reference>*
Possibly multiple <output> elements (multiple targets)	OK: **separated <coupling>"s" with the same <couplingSource> but having different <couplingTarget>"s**" involving the same "transient" in the connection bellow.
<lag>	OK: **@timeLag of a <connection> or a <coupling>** by inheritance
@transi_out_name	TODO-OASIS: will be **reconstructed based on the @shortName of the <componentProperty>** "transient" and the number of connections this transient is involved in as a <connectionTarget>.

OASIS4 parameter	CIM
<source_transformation> <scattering> <add_scalar> <mult_scalar> *<u>Note:</u> OASIS may automatically recognize <scattering> as a transformation to be performed on the source but not the other ones (<add_scalar> and <mult_scalar>)*	OK-ADDED: each will be expressed as a **<coupling>/<couplingSource>/<couplingProperty> OR a <connection>/<connectionSource>/<connectionProperty>** (cf. **EndPoint** type)
Possible values: « taverage», « accumul »	OK: will be specified as **<timeTransformation> of a <connection> or <coupling>** TODO-OASIS: The values for time operations are distinct according whether they occur on the source or on the target. OASIS will perform an automatic analysis of the value and assign **the time_operation to the right place (source our target)**
<statistics> <masked_points> <notmasked_points>	OK-ADDED: each will be expressed as a **<coupling>/<couplingSource>/<couplingProperty> OR a <connection>/<ConnectionSource>/<connectionProperty>** (cf. **EndPoint** type)
<debug_mode>	OK: will be expressed as a **<couplingSource>/<couplingProperty> OR <connectionSource>/<connectionProperty>**
<physics> <valid_min> <valid_max>	OK-ADDED: **@validMin**, **@validMax** attached **to the <value> of a <componentProperty>**. (<value> is of type <PropertyType>) *<u>Note:</u> This would be the equivalent of CF. @valid_min, @valid_max*
<physical units>	OK: **@units in the <componentProperty>** “transient”
<nbr_bundles>	TODO-OASIS: this will be **determined by the number of <componentProperty> in the <componentProperty> involved in the coupling** (automatic detection) *<u>Note:</u> we decided not to add an attribute @nbr_bundles in <componentProperty>*

OASIS4 parameter	CIM
@transient_type *Possible values: "single", "bundle"*	TODO-OASIS: this will be be **determined by "nbr_bundles"** (automatic detection)
<numerics> *Possible values: "real", "double", "integer"*	OK-ADDED : @numericalType of <value> in <componentProperty>
<file> *(details)*	OK : corresponds to **<file>** of type **<dataObject>**
<name>	OK: **<dataObject>/<storage>/<fileStorage>/@fileName**
<suffix>	As a <dataProperty> of a <dataObject>
<io_mode> *Possible values : iosingle, parallel, distributed*	OK: will be expressed as a **<dataProperty> of <dataObject>** with name&value.
<packing> <scaling> <adding> <fill_value>	TO-ADD: @**packing,** @**scale_factor,** @**add_offset,** @_**Fillvalue** as CF attributes of a <dataObject> OK: For the moment we use **<dataProperty>** Allyn had added for this purpose. *Note: To be discussed with dataPackage guys whether it would be useful to add a <dataProcess> element containing @packing, @scale_factor, @add_offset, @_FillValue within a <dataObject>*

Appendix B: file change list

The following section lists the OASIS4 source files that were added, removed or simply modified for supporting the CIM. Note that some files were removed for simplification only. Their content was transferred in another source file; no source code was actually removed.

1. General coupling data configuration
   - Modified source files:
     - In module OASIS4: prismdrv_set_scc_info.F90, Makefile
   - Added source file :
     - In library Common_oa4: psmile_cim.F90
2. Detailed coupling data configuration
   - Modified source files:
     - In module OASIS4: prismdrv_get_udef_transients.F90, prismdrv_init_smioc_struct.F90, prismdrv.F90
     - In library Common_oa4: psmile_smioc_interface.F90, get_smioc_numbers.F90, get_transi_details.F90, get_grids_details.F90
     - In library PSMILe_oa4: psmile_init_mpi1.F90, psmile_smioc_init.F90
   - Removed source files :
     - In module OASIS4: prismdrv_get_smioc_file_name.F90
     - In library Common_oa4:get_smioc_grids_transi_nb.F90, get_smioc_transi_numbers.F90
   - Added source files :
     - In module OASIS4: prismdrv_get_comp_names.F90
     - In library Common_oa4: get_regrid_details.F90, get_cnct_source_details.F90, get_cnct_target_details.F90
3. Low level functions (XML parsing)
   - Modified source files:
     - In library Common_oa4:
       - sasa_c_f90.c
       - sasa_c_f90.h
       - sasa_c_xml.c
       - Makefile