



a) Listing Python program 'make_all_metadata.py'

```
#!/usr/bin/python
#
#
# New Python Version to create all XML from cera metadata
#
# Hans Ramthun
# 2007/2008
# C3Grid project
# ISO/DIF XML metadata generation
#
import os
import sys
#import tempfile
#import re

sys.path.append("/opt/c3grid/webservice/gen1")
from c3webservice import DataAccessService
version="1.3, 2007/2008"

class CreateISOXML:

    def __init__(self, props=None):
        ''' init programm, set some basic parameter '''
        self.basedir = str(os.environ.get('C3GRID_ROOT'))
        self.modulesdir=str(os.environ.get('MODULESHOME'))
        self.oaidir = self.basedir+"/oai"
        try:
            # version exists
            self.version=version
        except NameError:
            # version doesn't exist
            self.version="1.3, 2007/2008"

        # output XML type
        self.typeXML='ISO'
        # request type
        self.req=''
        # provider
        self.prov=''
        # project acronym in CERA
        self.proj=''
        # experiment acronym in CERA
        self.ac=''
        # type of (data) file
        self.ftype=''
        # prefix
        self.prefix=''
        # make all files new if new=1!
        # default: don't touch them
        self.new=''
        self.newC3=''
        self.infile=''
        try:
```



```
self.data=DataAccessService(None)
except IOError, e:
    print 'Error: '+e.value
self.filenamePrefix="/xml_iso_test_"
self.help="Create ISO XML file, Version "+self.version+"\n\n usage:
[python] make_all.py \n\n -file 'properties file' | -type '[ISO]/DIF' 'type
of output XML'\n
-prov 'wdcc/ff/gkss'\n
projectname'\n
file prefix \n
files) / 'test' (create files in test directory)\n
-newC3 'new' (create new all C3 projects from internal list)\n"
```

```
def _GetSqlStr(self, req, proj, ac, ftype):
    if req=='exp':
        _retVal="distinct cera2.flat_entry.entry_id from
CERA2.FLAT_CAMPA,cera2.flat_entry where CERA2.FLAT_CAMPA.PROJECT_ACRONYM LIKE
'%" +proj+"%" and cera2.flat_entry.entry_id=CERA2.FLAT_CAMPA.entry_id and
cera2.flat_entry.entry_type='experiment' and cera2.flat_entry.entry_acronym
like '%" +ac+"%' and not cera2.flat_entry.entry_acronym like '%_JSB_%' and not
cera2.flat_entry.entry_acronym like '%_PRE_%' and not
cera2.flat_entry.entry_acronym like '%_20C3M_%' and not
cera2.flat_entry.entry_acronym like '%_C20SA_%' and not
cera2.flat_entry.entry_acronym like '%_CONST%' and not
cera2.flat_entry.entry_acronym like '%_20CMS%"
        # only with complete access permission!!
        #_retVal="distinct cera2.flat_entry.entry_id from
CERA2.FLAT_CAMPA,cera2.flat_entry,CERA2.flat_conne where
CERA2.FLAT_CAMPA.PROJECT_ACRONYM LIKE '%" +proj+"%" and
cera2.flat_entry.entry_id=CERA2.FLAT_CAMPA.entry_id and
cera2.flat_entry.entry_type='experiment' and cera2.flat_entry.entry_acronym
like '%" +ac+"%' and not cera2.flat_entry.entry_acronym like '%_JSB_%' and not
cera2.flat_entry.entry_acronym like '%_PRE_%' and not
cera2.flat_entry.entry_acronym like '%_20C3M_%' and not
cera2.flat_entry.entry_acronym like '%_C20SA_%' and not
cera2.flat_entry.entry_acronym like '%_CONST%' and not
cera2.flat_entry.entry_acronym like '%_20CMS%' and
CERA2.flat_conne.ID_G=cera2.flat_entry.entry_id and
cera.Check_Perm.check_permit_4dsname(CERA2.flat_conne.ACRO_S)=0"
        #_retVal="distinct CERA2.flat_conne.ID_G from CERA2.flat_conne where
cera.Check_Perm.check_permit_4dsname(CERA2.flat_conne.ACRO_S)=0 and
CERA2.flat_conne.ID_G in (select distinct cera2.flat_entry.entry_id from
CERA2.FLAT_CAMPA,cera2.flat_entry where CERA2.FLAT_CAMPA.PROJECT_ACRONYM LIKE
'%" +proj+"%" and cera2.flat_entry.entry_id=CERA2.FLAT_CAMPA.entry_id and
cera2.flat_entry.entry_type='experiment' and cera2.flat_entry.entry_acronym
like '%" +ac+"%' and not cera2.flat_entry.entry_acronym like '%_JSB_%' and not
cera2.flat_entry.entry_acronym like '%_PRE_%' and not
cera2.flat_entry.entry_acronym like '%_20C3M_%' and not
cera2.flat_entry.entry_acronym like '%_C20SA_%' and not
cera2.flat_entry.entry_acronym like '%_CONST%' and not
cera2.flat_entry.entry_acronym like '%_20CMS%)"
    else:
        if ftype=='':
```



```
        _retVal="distinct ID_S entry_id from CERA2.FLAT_CONNE where ID_G in
(select distinct cera2.flat_entry.entry_id from
CERA2.FLAT_CAMPA,cera2.flat_entry where CERA2.FLAT_CAMPA.PROJECT_ACRONYM LIKE
'"+proj+"%' and cera2.flat_entry.entry_id=CERA2.FLAT_CAMPA.entry_id and
cera2.flat_entry.entry_type='experiment' and cera2.flat_entry.entry_acronym
like '"+ac+"%' and not cera2.flat_entry.entry_acronym like '%_JSB_%' and not
cera2.flat_entry.entry_acronym like '%_PRE_%' and not
cera2.flat_entry.entry_acronym like '%_20C3M_%' and not
cera2.flat_entry.entry_acronym like '%_C20SA_%' and not
cera2.flat_entry.entry_acronym like '%_CONST%' and not
cera2.flat_entry.entry_acronym like '%_20CMS%')"
```

```
        #_retVal="distinct ID_S entry_id from CERA2.FLAT_CONNE where ID_G in
(select distinct cera2.flat_entry.entry_id from
CERA2.FLAT_CAMPA,cera2.flat_entry where CERA2.FLAT_CAMPA.PROJECT_ACRONYM LIKE
'"+proj+"%' and cera2.flat_entry.entry_id=CERA2.FLAT_CAMPA.entry_id and
cera2.flat_entry.entry_type='experiment' and cera2.flat_entry.entry_acronym
like '"+ac+"%' and not cera2.flat_entry.entry_acronym like '%_JSB_%' and not
cera2.flat_entry.entry_acronym like '%_PRE_%' and not
cera2.flat_entry.entry_acronym like '%_20C3M_%' and not
cera2.flat_entry.entry_acronym like '%_C20SA_%' and not
cera2.flat_entry.entry_acronym like '%_CONST%' and not
cera2.flat_entry.entry_acronym like '%_20CMS%') and
cera.Check_Perm.check_permit_4dsname(CERA2.flat_conne.ACRO_S)=0"
        else:
            _retVal="distinct entry_id,target from cera2.external_pointer where
ENTRY_ID in (select ID_S from CERA2.FLAT_CONNE where ID_G in (select distinct
cera2.flat_entry.entry_id from CERA2.FLAT_CAMPA,cera2.flat_entry where
CERA2.FLAT_CAMPA.PROJECT_ACRONYM LIKE '"+proj+"%' and
cera2.flat_entry.entry_id=CERA2.FLAT_CAMPA.entry_id and
cera2.flat_entry.entry_type='experiment' and cera2.flat_entry.entry_acronym
like '"+ac+"%' and not cera2.flat_entry.entry_acronym like '%_JSB_%' and not
cera2.flat_entry.entry_acronym like '%_PRE_%' and not
cera2.flat_entry.entry_acronym like '%_20C3M_%' and not
cera2.flat_entry.entry_acronym like '%_C20SA_%' and not
cera2.flat_entry.entry_acronym like '%_CONST%' and not
cera2.flat_entry.entry_acronym like '%_20CMS%')"
```

```
        return _retVal
```

```
def parseCmdLine(self, argv):
    try:
        argc=len(argv)
        #print argc
        if argc!=1:
            for i in range(0,argc-1):
                #print i
                #print argv[i]
                if argv[i]=='-type':
                    i+=1
                    if i!=argc:
                        self.typeXML=argv[i]
                elif argv[i]=='-req':
                    i+=1
                    if i!=argc:
```



```
        self.req=argv[i]
    elif argv[i]=='-prov':
        i+=1
        if i!=argc:
            self.prov=argv[i]
    elif argv[i]=='-proj':
        i+=1
        if i!=argc:
            self.proj=argv[i]
    elif argv[i]=='-ac':
        i+=1
        if i!=argc:
            self.ac=argv[i]
    elif argv[i]=='-prefix':
        i+=1
        if i!=argc:
            self.prefix=argv[i]
    elif argv[i]=='-file':
        i+=1
        if i!=argc:
            self.infile=argv[i]
    elif argv[i]=='-new':
        self.new='new'
    elif argv[i]=='-newC3':
        self.newC3='new'
    elif argv[i]=='-help':
        print self.help
        os._exit(-1)
    #else:
        #print str(i)+"/"+argv[i]
        #print argc
        #if i==argc-1:
            #print self.help
            #os._exit(-2)
    else:
        print self.help
        os._exit(-2)
except IOError, e:
    print 'Error: '+e.value

def CreateFromIniFileXMLFile(self, props=None):
    ''' create ISO XMLs from input properties file '''
    #print "hello world"
    try:
        fp=open(self.infile,"rb")
        lines=fp.readlines()
        fp.close()
        lineno=0
        i = iter(lines)
        self.proj=''
        for line in i:
            lineno += 1
            line = line.strip()
            #print line
```



```
# Skip null lines
if not line: continue
# Skip lines which are comments
if line[0] == '#': continue
arr=line.split(',')
self.new=='
for k in range(0,len(arr)):
    #print arr[k].split('=') [1]
    if arr[k].split('=') [0]=='type':
        self.typeXML=arr[k].split('=') [1]
    elif arr[k].split('=') [0]=='req':
        self.req=arr[k].split('=') [1]
    elif arr[k].split('=') [0]=='prov':
        self.prov=arr[k].split('=') [1]
    elif arr[k].split('=') [0]=='proj':
        self.proj=arr[k].split('=') [1]
    elif arr[k].split('=') [0]=='ac':
        self.ac=arr[k].split('=') [1]
    elif arr[k].split('=') [0]=='new':
        self.new=arr[k].split('=') [1]
if self.typeXML=='ISO':
    self.prefix='de.dkrz.wdcc.iso'
elif self.typeXML=='DIF':
    self.prefix='de.dkrz.wdcc.dif'
else:
    self.prefix='de.dkrz.wdcc.iso'
#print self.req+", "+self.prov+", "+self.proj+", "+self.ac
self.CreateXMLFile()
except IOError, e:
    print e.value

def CheckFlatFile(self, id):
    ''' check permission of datasets for CERA experiment'''
    sqlstr="distinct id_S from CERA2.flat_conne where
CERA2.flat_conne.ID_G="+id
    retVal=self.data.loadCERA(sqlstr)
    arr=retVal.split()
    arrlen=len(arr)
    ##result="1"
    if arrlen!=0:
        for i in range(0,arrlen):
            sqlstr="distinct entry_id from CERA2.external_pointer where
CERA2.external_pointer.entry_id="+arr[i]
            ##print sqlstr
            retVal=self.data.loadCERA(sqlstr)
            if retVal==arr[i]:
                return '0'
            else:
                return '1'
            #if retVal==arr[i]: # and id!='2161117':
            # return "0"
    #return "1"
```



```
def CheckCeraPermission(self, id):
    ''' check permission of datasets for CERA experiment'''
    if self.CheckFlatFile(id)=='0':
        return "0"
    sqlstr="distinct ACRO_S from CERA2.flat_conne where
CERA2.flat_conne.ID_G="+id
    retVal=self.data.loadCERA(sqlstr)
    arr=retVal.split()
    arrlen=len(arr)
    ##result="1"
    if arrlen!=0:
        for i in range(0,arrlen):
            retVal=self.data.loadCERA3(arr[i])
            print id, arr[i], retVal
            if retVal=='OK': # and id!='2161117':
                return "0"
    return "1"

def CreateXMLFile(self, props=None):
    ''' get id list from cera and create ISO XML files'''

    if self.proj!='':
        sqlstr=''
        retVal=''

        if self.new=='new':
            self.filenamePrefix="/xml_iso_"
        else:
            self.filenamePrefix="/xml_iso_test_"
        outputDir=self.oaidir+self.filenamePrefix+self.prov+"_"+self.req
        if os.path.exists(outputDir)==0:
            os.mkdir(outputDir)
        outputFile=self.prov+"_"+self.typeXML+"_"+self.req+"_"+self.proj
        if len(self.ac)!=0:
            outputFile=outputFile+"_"+self.ac
            outputFile=outputFile+".inp"
            fptmp=open(outputDir+"/"+outputFile, 'wb')
            #print self.req+"/"+self.proj+"/"+self.ac+"/"+self.ftype
            sqlstr=self._GetSqlStr(self.req,self.proj,self.ac,self.ftype)
            ##print sqlstr
            # get the id list from CERA
            retVal=self.data.loadCERA(sqlstr)
            fptmp.write(retVal)
            fptmp.close
            #print sqlstr
            arr=retVal.split()
            #print arr
            arrlen=len(arr)
            if arrlen!=0:
                for i in range(0,arrlen):
```



```
filename=self.prefix+arr[i]+".xml"
permission=self.CheckCeraPermission(arr[i])
print "count is: "+str(i+1)+" from "+str(arrlen)+"", id="+arr[i]+",
output is: "+outputDir+"/"+filename+" permission is: "+permission
if permission=='0' and (self.new=='new' or self.new=='test'):
    try:
        #print "ID: "+arr[i]
        retVal=self.data.loadXSQL(arr[i])
        fptmp=open(outputDir+"/"+filename,'wb')
        fptmp.write(retVal)
        fptmp.close
    except IOError, e:
        retVal=arr[i]+" error: "+e.value
    else:
        retVal=arr[i]
else:
    print "No entry found for project: '"+self.proj+'"'
else:
    print self.help
    os._exit(-10)
```

```
if __name__=="__main__":
    """ base programm for C3Grid stage file processing """

    # create object for stage file ISO XML file creation from CERA id list
    p2=CreateISOXML()
    # parse command line
    p2.parseCmdLine(sys.argv)
    # create files with paresd parameter
    #print "infile: "+p2.infile
    if p2.infile!='':
        # read parmeter from properties file
        p2.CreateFromIniFileXMLFile()
    elif p2.newC3=='new':
        # only experiments
        p2.req="exp"
        #####
        #
        # WDCC
        #
        #####
        p2.prov="wdcc"
        #####
        # project is IPCC_HH
        p2.proj="IPCC_HH"
        #####
        # ACRO = EH5_T63L31_OM
        p2.ac="EH5_T63L31_OM"
        p2.CreateXMLFile()
        # MPI-OM, ACRO = OM-GR1.5_EH5-T63
        p2.ac="OM-GR1.5_EH5-T63"
        p2.CreateXMLFile()
        # MPI-OM, ACRO = OM-GR1.5L40_EH5-T63
        p2.ac="OM-GR1.5L40_EH5-T63"
```



```
p2.CreateXMLFile()
#####
#
# Flat Files
#
#####
p2.prov="ff"
#####
# project is IPCC_HH
p2.proj="IPCC_HH"
#####
# ACRO = EH5T
#####
p2.ac="EH5T"
p2.CreateXMLFile()
#####
#PROJ=DRAKKAR
#####
p2.proj="DRAKKAR"
p2.ac=""
p2.CreateXMLFile()
#####
#
# GKSS experiments
#
#####
p2.prov="gkss"
p2.ac=""
#####
# project is SO&P
#####
p2.proj="SOaP"
p2.CreateXMLFile()
#####
# project is MIDHOL
#####
p2.proj="MIDHOL"
p2.CreateXMLFile()
#####
# project is DEKLIM
#####
p2.proj="DEKLIM"
p2.CreateXMLFile()
#####
# project is GLIMPSE
#####
p2.proj="GLIMPSE"
p2.CreateXMLFile()
else:
p2.CreateXMLFile()
```



b) Listing of Python program c3webservice.py

```
#!/usr/bin/python
#
# utility module for the C3Grid webservice data stage interface
#
# Hans Ramthun
# DKRZ/MPI-M v 1.1, 2007,2008
#
#
#
# standard libraries
import os
import sys
import string
import tempfile
import commands
import time

# additional libraries
# XML support
from lxml import etree
# retrieving data with http/https
from urllib2 import Request, urlopen
#from _urllib2 import Request, urlopen
#import socket
from StringIO import StringIO

#from xml.sax import saxutils

# set to 'DEBUG' for logging
logtypeVal='DEBUG'
DEFAULT_LOGTYPE="ERROR"

class DataLogging:
    """ working on ISO XML metadata, create and update """

    def __init__(self, props=None):
        """ initialize DataAccessService object """
        try:
            self.logtype=os.environ.get('C3LOGTYPE')
        except:
            self.logtype=DEFAULT_LOGTYPE
        self._keys = {}
        self.error = -1
        self.stdVal = 0

    def printf(self,str,type):
        """ encapsulated print function for all modules """

        if self.logtype==type or self.logtype==DEFAULT_LOGTYPE:
            print ('type='+type+', '+str)
```



```
class DataAccessService:
    """ working on ISO XML metadata, create and update """

    def __init__(self, props=None):
        """ initialize DataAccessService object """
        self.logtype=os.environ.get('C3LOGTYPE')
        self.log=DataLogging()
        self._keys = {}
        self.error = -1
        self.stdVal = 0

        # set timeout in seconds for urlopen
        #timeout = 60
        #socket.setdefaulttimeout(timeout)
        # default parameter MAD CERA SQL servlet server
        self.set_param("c3grid.sql.servlet","http://cera-
www.dkrz.de/ceraMeta/SelectGeneric?fo=s&q=")

        # default parameter MAD CERA SQL servlet server
        self.set_param("c3grid.sqlMinMax.servlet","http://cera-
www.dkrz.de/ceraMeta/SelectMaxMin?fo=s&sep=&t=")

        # default parameter C3 user for permission servlet
        self.set_param("c3grid.permission.servlet","http://cera-
www.dkrz.de/ceraMeta/SelectPea?pass=370c17141c5a5058&user=375b&fo=s&ac=")

        # default parameter OAI server
        self.set_param("c3grid.oai.server","http://anticyclone.dkrz.de:8080/oai/p
rovider?verb=GetRecord&metadataPrefix=iso&identifier=")

        # default parameter XSQL server for list and XML create
        self.set_param("c3grid.xsql.servlet","http://anticyclone.dkrz.de:8080/xsq
l/cera_map_iso.xsql?id=")
        #self.set_param("c3grid.xsql.server.list","http://uranus.dkrz.de:8080/oai
/provider/xsql/cera_map_list.xsql")

        # default parameter CF server
        self.set_param("c3grid.cf.xml.url","http://cf-
pcmdi.llnl.gov/documents/cf-standard-names/7/cf-standard-name-table.xml")

        # prefix for OAI server
        self.set_param("c3grid.isoprefix","de.dkrz.wdcc.iso")

        #print ("Init function in object: "+self.__class__.__name__)
        self.log.printf("Init function in object:
"+self.__class__.__name__,logtypeVal)

    def get_all_keys(self):
        """ get the internal dictionary key list """
        return self._keys.keys()
```



```
def get_all_values(self):
    """ get the internal dictionary value list """
    return self._keys.values()

def get_param(self, key):
    """ return a class parameter """
    if (key in self._keys.keys())==True:
        retVal=self._keys[key]
    else:
        retVal=''
    return retVal

def set_param(self, key, value):
    """ set a class parameter, create if not exists """
    #if self.logtype=='DEBUG':
    # print ("Set key: "+key+" to value: "+value+"\n")
    self._keys[key]=value.strip()

def loadXML(self, address, file):
    """ Load xml document conforming to ISO 19139
    iso 19139 xsd conformance checking is done !
    Use validate() for xsd validation """
    if address =='localhost':
        self.c3_tree = etree.parse(file)
    elif address.startswith('http'):
        _result = self.getrecord_httpserver(address,file)
        #_result = _response.read()
        # to do: adapt to parse not a file .. !!!
        f=StringIO(_result)
        self.c3_tree = etree.parse(f)
    elif 1:
        raise IOError,' server type not supported'
        exit()
    self.log.printf(' '+file+' ..... parsed',logtypeVal)

def parseSQLString(self, sqlstr):
    """ replace of special characters """
    sqlstr=sqlstr.replace('%','%25')
    sqlstr=sqlstr.replace('&','%26')
    sqlstr=sqlstr.replace(' ','%20')
    sqlstr=sqlstr.replace('(','%28')
    sqlstr=sqlstr.replace(')','%29')
    sqlstr=sqlstr.replace(',')','%2C')
    sqlstr=sqlstr.replace('%20%20','%20')
```

Attention: no



```
return sqlstr

def loadCERA(self, sqlstr):
    """ call a CERA SQL request with the MAD servlet parameter is the SQL
    string, result is a string or a list """
    self.log.printf("loadCERA' Input SQL string is: "+sqlstr,logtypeVal)
    sqlstr=self.parseSQLString(sqlstr)
    self.log.printf("Complete Servlet Server Call is:
"+self.get_param('c3grid.sql.servlet')+sqlstr+'\n',logtypeVal)
    _retVal=self.getrecord_httpserver(self.get_param('c3grid.sql.servlet'),sq
lstr)
    return _retVal.strip()

def loadCERA2(self, sqlstr):
    """ call a CERA SQL request with the MAD servlet parameter is the SQL
    string, result is a string or a list """
    self.log.printf("loadCERA2' Input SQL string is: "+sqlstr,logtypeVal)
    sqlstr=self.parseSQLString(sqlstr)
    self.log.printf("Complete Servlet Server Call is:
"+self.get_param('c3grid.sql.servletMinMax')+sqlstr+'\n',logtypeVal)
    _retVal=self.getrecord_httpserver(self.get_param('c3grid.sql.servletMinMa
x'),sqlstr)
    return _retVal.strip()

def loadCERA3(self, sqlstr):
    """ call a CERA SQL request with the MAD servlet parameter is the SQL
    string, result is a string or a list """
    self.log.printf("loadCERA3' Input SQL string is: "+sqlstr,logtypeVal)
    sqlstr=self.parseSQLString(sqlstr)
    self.log.printf("Complete Servlet Server Call is:
"+self.get_param('c3grid.permission.servlet')+sqlstr+'\n',logtypeVal)
    _retVal=self.getrecord_httpserver(self.get_param('c3grid.permission.servl
et'),sqlstr)
    return _retVal.strip()

def loadXSQL(self, sqlstr):
    """ call a CERA XSQL request at XSQL servlet server (actual address is:
http://uranus.dkrz.de:8080/oai/provider/xsql) """
    _retVal=self.getrecord_httpserver(self.get_param('c3grid.xsql.servlet'),s
qlstr)
    #_retVal=_response.read()
    #f=StringIO(_result)
    #self.c3_tree=etree.parse(f)
    return _retVal.strip()
```



```
def getUrlInfo(self):
    return "#Test" #+self.info1 #+"#" +self.info2+"#"
#self.infoStr1+"#" +self.infoStr2

def getItems(self):
    return self.items

def getRes(self):
    return self.res2

def getrecord_httpserver(self, server, identifier):
    """ retrieve data from a http server """
    req_string = server+identifier
    self.log.printf('urlopen-request-string is: '+req_string+'\n', logtypeVal)
    ##print 'urlopen-request-string is: '+req_string
    _response=""
    try:
        req = Request(req_string)
        urlfile = urlopen(req)
        _response=urlfile.read()
        urlfile.close()
    except IOError, e:
        if hasattr(e, 'reason'):
            self.log.printf('Server connect error: We failed to reach a
server.', logtypeVal)
            self.log.printf('Your http request was: '+req_string, logtypeVal)
            self.log.printf('Reason: '+e.reason, logtypeVal)
            sys.exit(-100)
        elif hasattr(e, 'code'):
            self.log.printf('Server error: The server couldn\'t fulfill the
request.', logtypeVal)
            self.log.printf('Request: '+req_string, logtypeVal)
            self.log.printf('Error code: '+e.code, logtypeVal)
            sys.exit(-200)
        #else:
        #_test = _response.read()
        #return "TEST" #_response
    #except HTTPError, e:
    #print ('Error2 code: '+e2.code)
    return _response

def transform(self, xsltfile, **params):
    """ transform of XML with XSLT stylesheet """
    _xslt = etree.parse(xsltfile)
    _transformer = etree.XSLT(_xslt)
    _result = transformer(self.c3_tree, **params)
    return _result

def getXSLTtree(self, doc):
    """ creates a tree from a transform xslt file """
    try:
```



```
        fp=open(doc,'rb')
        result_tree=etree.parse(fp)
        fp.close()
        result = etree.XSLT(result_tree)
    except IOError, e:
        self.log.printf('Could not open file: '+doc,logtypeVal)
        self.log.printf('Error code: '+e.code,logtypeVal)
        exit()
    return result

def _function(self, name, value, param=None):
    """ internal function library to provide several functions """

    if name == 'getISOXML':
        self.loadXML(self.get_param('c3grid.oai.server'),value)
        self.log.printf("Complete OAI server get call is:
"+self.get_param('c3grid.oai.server')+value+'\n',logtypeVal)
        tempfilename=tempfile.mktemp()
        fptmp=open(tempfilename,'wb')
        transform=self.getXSLTtree(self.get_param('c3grid.scriptMetaDataAccess'
)) # self.metadataAccessScript
        result_tree=transform(self.c3_tree)
        fptmp.write(str(result_tree))
        fptmp.close
        # return the file(name) to access the output
        self.log.printf("Complete OAI server output is in:
"+tempfilename+'\n',logtypeVal)
        retVal=tempfilename

    else:
        retVal=self.error
    return retVal
```



c) Example property file for process control

```
#
# property file to control ISO/DIF XML generation with make_all_metadata.py
# call: 'make_all_metadata.py -type ISO -file c3grid.metadata.xml.properties'
# 2007/2008, Hans Ramthun
#
# syntax of a line is:
# new=new=overwrite existing files/test=create only test file
# type=XML output type:ISO/DIF
# req=request:exp([data] is actual not supported)
# prov=provider name:wdcc/ff/gkss
# proj=cera project name:IPCC_HH (e.g.)
# [ac=cera acronym(if available) or nothing (or space)]
#
#
# CERA IPCC data
#new=new,type=ISO,req=exp,prov=wdcc,proj=IPCC_HH,ac=EH5_T63L31_OM
#new=new,type=ISO,req=exp,prov=wdcc,proj=CLM,ac=
#new=test,type=ISO,req=exp,prov=wdcc,proj=ERA15,ac=
#new=test,type=ISO,req=exp,prov=wdcc,proj=ERA40,ac=
#new=test,type=ISO,req=exp,prov=wdcc,proj=REMO-UBA,ac=
#new=test,type=ISO,req=exp,prov=wdcc,proj=NCEP,ac=
#
# CERA MPI-OM data
#new=new,type=ISO,req=exp,prov=wdcc,proj=IPCC_HH,ac=OM-GR1.5_EH5-T63
#new=new,type=ISO,req=exp,prov=wdcc,proj=IPCC_HH,ac=OM-GR1.5L40_EH5-T63
#
#REMO-UBA
#new=test,type=ISO,req=exp,prov=ff,proj=REMO_UBA,ac=
#
# GKSS archive data (at DKRZ)
#
## o.k.
#new=test,type=ISO,req=exp,prov=gkss,proj=SOaP,ac=
#
## o.k.
#new=test,type=ISO,req=exp,prov=gkss,proj=MIDHOL,ac=
#
## 4 o.k., 1 not o.k.
##new=test,type=ISO,req=exp,prov=gkss,proj=GLIMPSE,ac=
#
#new=new,type=ISO,req=exp,prov=gkss,proj=HOAPS,ac=HOAPS3
#
#new=test,type=ISO,req=exp,prov=gkss,proj=DEKLIM,ac=
#
# other archive data (at DKRZ)
#
# IPCC
#new=new,type=ISO,req=exp,prov=ff,proj=IPCC_HH,ac=EH5T
#
# IFM-GEOMAR NEMO/OPA
#new=test,type=ISO,req=exp,prov=ff,proj=DRAKKAR,ac=
```



d) XSL(T) Stylesheet for output control

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xsl:stylesheet [ <!ENTITY % HTMLlat1 PUBLIC
    "-//W3C//ENTITIES Latin 1 for XHTML//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent"> ]>
<xsl:stylesheet
    version="1.0"
    xmlns="http://www.isotc211.org/2005/gmd"
    xmlns:iso="http://www.isotc211.org/2005/gmd"
    xmlns:oai="http://www.openarchives.org/OAI/2.0/"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:gco="http://www.isotc211.org/2005/gco"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:date="http://exslt.org/dates-and-times"
    xmlns:func="http://exslt.org/functions"
    xmlns:madxslt="http://www.oracle.com/XSL/Transform/java/mad.xslt.MADXSLTServlet"
    xmlns:madtools="http://www.oracle.com/XSL/Transform/java/mad.cera.tools.UrnTool"
    extension-element-prefixes="date"
    exclude-result-prefixes="xsl madxslt madtools date func">

    <xsl:import href="./exslt/add/date.add.xsl"/>
    <xsl:import href="./exslt/difference/date.difference.xsl"/>

    <xsl:output version="1.0" encoding="UTF-8" method="xml" indent="yes" media-type="text/xml"/>
    <xsl:strip-space elements="MD_Metadata"/>

    <xsl:param name="request" select="''"/>
    <xsl:param name="metadataPrefix" select="'iso'"/>
    <xsl:param name="ffname" select="''" />
    <xsl:param name="debug" select="''" />

    <!--
    xmlns:madxslt="http://www.oracle.com/XSL/Transform/java/mad.servlets.MADXSLTServlet"

        xmlns:ora-exslt-
date="http://www.oracle.com/XSL/Transform/java/org.apache.xalan.lib.ExsltDate
time"
        extension-element-prefixes="date"
        xmlns:func="http://exslt.org/functions"

xmlns:madxslt="http://www.oracle.com/XSL/Transform/java/mad.servlets.MADXSLTServlet"
    <xsl:import href="./exslt/date/date.date.xsl"/>
    <xsl:import href="./exslt/year/date.year.xsl"/>

        <xsl:result-document
            href="../output/CD{position()}_{current-
group()/COUNTRY}.xml"
```



```
        format="iso-format">
<xsl:call-template name="dateMetaVersion"/>
</xsl:result-document>

<xsl:character-map name="cm1">
  <xsl:output-character character="#160;" string="&nbsp;"/>
  <xsl:output-character character="#8212;" string=""/>
</xsl:character-map>
<xsl:output version="1.0" encoding="iso-8859-1" method="xml" indent="yes"
media-type="text/xml" use-character-maps="cm1" />
-->

<xsl:variable name="rev">1.0</xsl:variable>

<xsl:variable name="schemaLoc">http://www.isotc211.org/2005/</xsl:variable>
<xsl:variable
name="codelistLoc">http://wis.wmo.int/2006/catalogues/</xsl:variable>
  <xsl:variable name="codelist"><xsl:value-of
select="$codelistLoc"/>gmxCodelists.xml</xsl:variable>
  <xsl:variable name="codelistML"><xsl:value-of
select="$codelistLoc"/>ML_gmxCodelists.xml</xsl:variable>

  <xsl:variable
name="verticalCRSTemplateURL">http://www.c3grid.de/files/schemas/C3grid-crs-
dict.xml</xsl:variable>
  <xsl:variable name="verticalElementPrefix">verticalCRS_</xsl:variable>
  <xsl:variable
name="metadataTemplateURL">http://c3grid.de/wiki/images/5/5b/C3-xml-template-
1-1.xml</xsl:variable>

  <xsl:variable name="distributorPrefix"></xsl:variable>
  <xsl:variable name="distributorPrefix2">wdcc</xsl:variable>

  <xsl:variable name="institutePrefix"><xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains(CONTACT_TYPE, 'metadata')]/INSTITUTE_
NAME"/></xsl:variable>
<!--
  <xsl:variable name="institutePrefix"><xsl:value-of
select="'MPI'"/></xsl:variable>
  <xsl:variable name="sitePrefix">de.dkrz.wdcc.</xsl:variable>
-->
  <xsl:variable name="sitePrefixWDCC">de.dkrz.wdcc.</xsl:variable>
  <xsl:variable name="sitePrefixMPIM">de.dkrz.mpim.</xsl:variable>
  <xsl:variable name="sitePrefixGKSS">de.dkrz.gkss.</xsl:variable>
  <xsl:variable name="sitePrefixIFM-GEOMAR">de.dkrz.ifm-geomar.</xsl:variable>
  <xsl:variable name="sitePrefix">
    <xsl:choose>
      <xsl:when test="contains($institutePrefix, 'Max-Planck-
Institut')"><xsl:value-of select="$sitePrefixMPIM"/></xsl:when>
      <xsl:when test="contains($institutePrefix, 'GKSS')"><xsl:value-of
select="$sitePrefixGKSS"/></xsl:when>
      <xsl:when test="contains($institutePrefix, 'IFM-GEOMAR')"><xsl:value-of
select="$sitePrefixIFM-GEOMAR"/></xsl:when>
```



```
<xsl:otherwise><xsl:value-of select="$sitePrefixWDCC"/></xsl:otherwise>
</xsl:choose>
</xsl:variable>

<xsl:variable name="ceraPrefix">cera</xsl:variable>

<xsl:variable name="organisationName">World Data Center for
Climate</xsl:variable>
<xsl:variable name="organisationMail">data@dkrz.de</xsl:variable>
<xsl:variable name="organisationURL">http://www.mad.dkrz.de/</xsl:variable>
<xsl:variable name="metadataName">ISO 19115</xsl:variable>
<xsl:variable name="metadataVersion">ISO 19139 / C3Grid Profile
V1.0</xsl:variable>
<xsl:variable name="category" select="'EARTH SCIENCE'"/>

<xsl:variable name="C3Grid_Models">C3Grid_models</xsl:variable>

<!-- Java Version
<xsl:variable name="providerEndpoint"><xsl:value-of
select="madxslt:testServlet('c3grid.provider.test.providerEndpoint')"/></xsl:
variable>
<xsl:variable name="workspaceDir"><xsl:value-of
select="madxslt:testServlet('c3grid.provider.test.workspaceDir')"/></xsl:vari
able>
-->
<xsl:variable name="webmdsEndpoint"><xsl:value-of select="'http://c3grid-
gt.e-technik.uni-
dortmund.de:8080/webmds/webmds?info=indexinfo'"/></xsl:variable>
<xsl:variable name="webmdsProtocol">http</xsl:variable>
<xsl:variable
name="webmdsName">C3Grid_ResourceInformationService</xsl:variable>
<xsl:variable name="webmdsDescr">Production System: DKRZ C3Grid Data
Request Interface Implementation</xsl:variable>

<xsl:variable name="webserviceEndpoint2"><xsl:value-of
select="'https://uranus.dkrz.de:9443/axis/services/C3ProviderSOAP'"/></xsl:va
riable>
<xsl:variable name="webserviceEndpoint1"><xsl:value-of
select="'http://anticyclone.dkrz.de:8080/axis/services/C3ProviderSOAP'"/></xs
l:variable>
<xsl:variable name="webserviceEndpoint"><xsl:value-of
select="'https://anticyclone.dkrz.de:9443/axis/services/C3ProviderSOAP'"/></x
sl:variable>
<xsl:variable name="webserviceProtocol">SOAP</xsl:variable>
<xsl:variable
name="webserviceName">C3Grid_DataProvider_Webservice</xsl:variable>
<xsl:variable name="webserviceDescr">C3Grid Data Provider Webservice at
DKRZ</xsl:variable>

<xsl:variable name="workspaceDir2"><xsl:value-of
select="'gsiftp://gridftp.dkrz.de/prj/bb0300/work/dms_workspace'"/></xsl:vari
able>
<xsl:variable name="workspaceDir"><xsl:value-of
select="'gsiftp://anticyclone.dkrz.de/opt/c3grid/dms_workspace'"/></xsl:varia
ble>
```



```
<xsl:variable name="workspaceProtocol">gsiftp</xsl:variable>
<xsl:variable name="workspaceName">C3Grid_Gsiftp_Data</xsl:variable>
<xsl:variable name="workspaceDescr">C3Grid Workspace at DKRZ</xsl:variable>

<xsl:variable name="C3GridDataAccessInfo"><xsl:value-of
select="'C3Grid_DataAccess_Information'"/></xsl:variable>

<xsl:variable name="MPI-OM">
  <xsl:choose>
    <xsl:when test="contains(/CERA2/ENTRY/ROWSET/ROW/ENTRY_ACRONYM, 'OM-
GR1.5_EH5-T63') or contains(/CERA2/ENTRY/ROWSET/ROW/ENTRY_ACRONYM, 'OM-
GR1.5L40_EH5-T63')">MPI-OM</xsl:when>
    <xsl:otherwise></xsl:otherwise>
  </xsl:choose>
</xsl:variable>>

<!-- problem to define as global variable!!
  <xsl:variable name="actDateTime" select="substring-
before(date:add(date:date-time(), '-PT0H'), 'Z')" />
  <xsl:variable name="actDate" select="substring-
before(date:add(date:date(), '-PT0H'), 'T')" />

  <xsl:variable name="actDateTime1">
    <xsl:call-template name="date:date-time"/>
  </xsl:variable>
  <xsl:variable name="actDateTime2">
    <xsl:call-template name="date:add">
      <xsl:with-param name="date-time" select="$actDateTime1" />
      <xsl:with-param name="duration" select="'-PT0H'" />
    </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="actDateTime1" select="date:date-time()"/>
  <xsl:variable name="actDateTime2">
    <xsl:call-template name="date:add">
      <xsl:with-param name="date-time" select="$actDateTime1" />
      <xsl:with-param name="duration" select="'-PT0H'" />
    </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="actDateTime" select="substring-
before($actDateTime2, 'Z')" />
  -->
  <xsl:variable name="actDateTime"
select="/CERA2/ACT_DATETIME/ROWSET/ROW/ACT_DATETIME" />
  <xsl:variable name="expSummary"><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/SUMMARY"/></xsl:variable>
  <xsl:variable name="entryAcro"><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_ACRONYM"/></xsl:variable>
  <xsl:variable name="projAcro"><xsl:value-of
select="/CERA2/CAMPA/ROWSET/ROW/PROJECT_ACRONYM"/></xsl:variable>

  <xsl:variable name="n_a" select="'n/a'"/>
  <xsl:variable name="n_f" select="'not filled'"/>
  <xsl:variable name="MyDummy">not known</xsl:variable>
  <xsl:variable name="delimiter">#</xsl:variable>
  <xsl:variable name="nonCF">nonCF</xsl:variable>
```



```
<xsl:variable name="isCF">CF-Standard Name</xsl:variable>

<xsl:variable name="debugInfo" select="'Debuginfo:'"/>
<xsl:variable name="debugInfoTmpl" select="'Debuginfo(name of template):'"/>

<xsl:variable name="eID"><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_ID"/></xsl:variable>
<xsl:variable name="fileID">
  <xsl:value-of select="$sitePrefix"/>
  <xsl:value-of select="$metadataPrefix"/>
  <xsl:value-of select="$eID"/>
  <xsl:if test="$ffname!=''">
    <xsl:value-of select="concat('-', $ffname)"/>
  </xsl:if>
</xsl:variable>
<xsl:variable name="pID"><xsl:value-of
select="/CERA2/CONNE_DS/ROWSET/ROW/ID_G"/></xsl:variable>
<xsl:variable name="parentID">
  <xsl:value-of select="$sitePrefix"/>
  <xsl:value-of select="$metadataPrefix"/>
  <xsl:value-of select="$pID"/>
</xsl:variable>

  <xsl:variable name="doiPrefix" select="'#DOI:'"/>
<xsl:variable name="doiSTR">
  <xsl:value-of select="/CERA2/IS_DOI_EXP/ROWSET/ROW/DOI_STR"/>
  <xsl:value-of select="/CERA2/IS_DOI_DS/ROWSET/ROW/DOI_STR"/>
</xsl:variable>

  <!-- REF_TYPE_ID=2000004 results in DOI experiments only -->
<xsl:variable name="doi_exp_id" select="'2000004'"/>
  <!-- REF_TYPE_ID=2000006 results in DOI datasets only -->
<xsl:variable name="doi_data_id" select="'2000006'"/>

  <xsl:variable name="data_compactURL" select="'http://cera-
www.dkrz.de/WDC/Climate/Compact.jsp?acronym='"/>

  <xsl:variable name="blackList" select="'EH5-T63L31_OM_1CO2_1_6H,EH5-
T63L31_OM_1CO2_2_6H'"/>
  <xsl:variable name="C3Grid_NO_ACCESS"
select="'data_temporary_not_available'"/>
  <xsl:variable name="modelList" select="'ECHAM4,ECHAM5,ORCA05,ECHO-
G,MPI-OM'"/>
  <xsl:variable name="GKSSProjLst" select="'MIDHOL,SOaP,DEKLIM,GLIPMSE'"/>

<xsl:variable name="dataType">
  <xsl:choose>
    <xsl:when test="( /CERA2/CHECK_EXT_EXP/ROWSET/ROW/CNT!=0) or
( /CERA2/CHECK_EXT_DS/ROWSET/ROW/CNT!=0) ">
      <xsl:value-of select="'DKRZ Archive Data'"/>
    </xsl:when>
    <xsl:when test="( /CERA2/CHECK_CERA_EXP/ROWSET/ROW/CNT!=0) or
( /CERA2/CHECK_CERA_DS/ROWSET/ROW/CNT!=0) ">
      <xsl:value-of select="'CERA Database Data'"/>
    </xsl:when>
  </xsl:choose>
</xsl:variable>
```



```
</xsl:choose>
</xsl:variable>

<!-- Template ISO experiment -->
<xsl:template name="cera_map_iso_exp">
  <MD_Metadata
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="{ $schemaLoc }gmd
{ $schemaLoc }gmd/metadataEntity.xsd"
    id="{ $fileID }">

    <fileIdentifier>
      <gco:CharacterString>
        <xsl:value-of select="$fileID"/>
      </gco:CharacterString>
    </fileIdentifier>

    <hierarchyLevel>
      <MD_ScopeCode codeList="{ $codelist }#MD_ScopeCode"
codeListValue="series">series</MD_ScopeCode>
    </hierarchyLevel>

    <hierarchyLevelName>
      <gco:CharacterString><xsl:value-of select="$projAcro"/>:<xsl:value-of
select="$entryAcro"/></gco:CharacterString>
    </hierarchyLevelName>

    <contact>
      <!-- general contacts for meta data provider -->
      <CI_ResponsibleParty>
        <xsl:call-template name="printProviderInfo">
          <xsl:with-param name="function" select="'metadata'"/>
        </xsl:call-template>
        <xsl:call-template name="printRole">
          <xsl:with-param name="role" select="'distributor'"/>
        </xsl:call-template>
      </CI_ResponsibleParty>
    </contact>

    <xsl:call-template name="dateMetaVersion"/>

    <xsl:element name="dataSetURI">
      <xsl:call-template name="dataSetURI"/>
    </xsl:element>

    <!-- for german description of names,... -->
    <xsl:call-template name="locale"/>

    <!-- actual not used
    <xsl:element name="spatialRepresentationInfo">
      <xsl:call-template name="spatialRepresentationInfo"/>
    </xsl:element>

    <xsl:element name="referenceSystemInfo">
      <xsl:call-template name="referenceSystemInfo"/>

```



```
</xsl:element>
-->

<!--
-->
<xsl:call-template name="identificationInfo"/>

<xsl:call-template name="contentInfo"/>

<xsl:element name="distributionInfo">
  <xsl:call-template name="distributionInfo"/>
</xsl:element>

<xsl:element name="dataQualityInfo">
  <xsl:call-template name="dataQualityInfo"/>
</xsl:element>

</MD_Metadata>

</xsl:template>

<!-- Template ISO dataset -->
<xsl:template name="cera_map_iso_ds">
  <MD_Metadata
    xmlns="http://www.isotc211.org/2005/gmd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="{ $schemaLoc }gmd
{ $schemaLoc }gmd/metadataEntity.xsd"
    id="{ $fileID }">

    <fileIdentifier>
      <gco:CharacterString>
        <xsl:value-of select="$fileID"/>
      </gco:CharacterString>
    </fileIdentifier>

    <parentIdentifier>
      <gco:CharacterString>
        <xsl:value-of select="$parentID"/>
      </gco:CharacterString>
    </parentIdentifier>

    <hierarchyLevel>
      <MD_ScopeCode codeList="{ $codelist }#MD_ScopeCode"
codeListValue="dataset">dataset</MD_ScopeCode>
    </hierarchyLevel>

    <xsl:variable name="exp" select="/CERA2/CONNE_DS/ROWSET/ROW/ID_G"/>
    <hierarchyLevelName>
      <gco:CharacterString><xsl:value-of
select="/CERA2/CAMPA_ALL/ROWSET/ROW[ENTRY_ID=$exp]/PROJECT_ACRONYM"/>:<xsl:va
lue-of select="$entryAcro"/></gco:CharacterString>
    </hierarchyLevelName>
```



```
<contact>
  <!-- general contacts for meta data provider, only party?
  Authors and investigators at point of contact
  -->
  <!-- block for contact1:TOP -->
  <CI_ResponsibleParty>
    <xsl:call-template name="printProviderInfo">
      <xsl:with-param name="function" select="'metadata'"/>
    </xsl:call-template>
    <xsl:call-template name="printRole">
      <xsl:with-param name="role">distributor</xsl:with-param>
    </xsl:call-template>
  </CI_ResponsibleParty>
</contact>

<xsl:call-template name="dateMetaVersion"/>

<xsl:element name="dataSetURI">
  <xsl:call-template name="dataSetURI"/>
</xsl:element>

<xsl:call-template name="locale"/>

<!-- erstmal weglassen
<xsl:element name="spatialRepresentationInfo">
  <xsl:call-template name="spatialRepresentationInfo"/>
</xsl:element>

<xsl:if test="$ffname=''">
  <xsl:element name="referenceSystemInfo">
    <xsl:call-template name="referenceSystemInfo"/>
  </xsl:element>
</xsl:if>
-->

<xsl:element name="identificationInfo">
  <xsl:call-template name="identificationInfoData"/>
</xsl:element>

<xsl:call-template name="contentInfoData"/>

<xsl:element name="distributionInfo">
  <xsl:call-template name="distributionInfoData"/>
</xsl:element>

<xsl:element name="dataQualityInfo">
  <xsl:call-template name="dataQualityInfoData"/>
</xsl:element>

  </MD_Metadata>
</xsl:template>

<xsl:template name="dateMetaVersion">
  <dateStamp>
```



```
<gco:DateTime>
  <xsl:value-of select="/CERA2/ENTRY/ROWSET/ROW/REVIEW_DATE_OAI"/>
</gco:DateTime>
</dateStamp>
<metadataStandardName>
  <gco:CharacterString>
    <xsl:value-of select="$metadataName"/>
  </gco:CharacterString>
</metadataStandardName>
<metadataStandardVersion>
  <gco:CharacterString>
    <xsl:value-of select="$metadataVersion"/>
  </gco:CharacterString>
</metadataStandardVersion>
</xsl:template>

<xsl:template name="locale">
  <xsl:if test="$debug='showTplNames'">
    <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="'locale'"/>
  </xsl:if>
  <locale>
    <!-- for german description of names,... -->
    <PT_Locale id="de">
      <languageCode>
        <LanguageCode codeList="{ $codelistML }#LanguageCode"
codeListValue="de">de</LanguageCode>
      </languageCode>
      <country>
        <Country codeList="{ $codelistML }#Country"
codeListValue="Germany">Germany</Country>
      </country>
      <characterEncoding>
        <MD_CharacterSetCode codeList="{ $codelistML }#MD_CharacterSetCode"
codeListValue="8859part1">8859part1</MD_CharacterSetCode>
      </characterEncoding>
    </PT_Locale>
  </locale>
</xsl:template>

<xsl:template name="dataSetURI">
  <gco:CharacterString>
    <xsl:choose>
      <xsl:when test="$doiSTR!=''">
        <xsl:value-of select="'[doi: '"/>
        <xsl:value-of select="$doiSTR"/>
        <xsl:value-of select="']'"/>
      <xsl:value-of select="' [urn:nbn:de:tib-'"/>
      <!-- calculateChecksum($doiSTR)
<xsl:value-of select="madtools:getResultURN()"/>
<xsl:value-of select="madtools:UrnTool.calculateChecksum($doiSTR)"/>
-->
      <xsl:value-of select="madtools:calculateChecksum($doiSTR)"/>
    </xsl:choose>
  </gco:CharacterString>
</xsl:template>
```



```
<xsl:value-of select="'']'"/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="$data_compactURL"/><xsl:value-of
select="$entryAcro"/>
  </xsl:otherwise>
</xsl:choose>
</gco:CharacterString>
</xsl:template>
```

```
<xsl:template name="printAuthor">
  <xsl:if test="$debug='showTplNames'">
    <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="'printAuthor'"/>
  </xsl:if>
  <xsl:value-of select="LAST_NAME"/>
  <xsl:if test="FIRST_NAME">
    <xsl:text>, </xsl:text>
    <xsl:value-of select="FIRST_NAME"/>
  </xsl:if>
  <xsl:if test="SECOND_NAME and not(contains(SECOND_NAME,$n_a))">
    <xsl:text>, </xsl:text>
    <xsl:value-of select="SECOND_NAME"/>
  </xsl:if>
</xsl:template>
```

```
<xsl:template name="printResponsibleParty">
  <individualName>
    <gco:CharacterString>
      <xsl:value-of select="PERSON_NAME"/>
    </gco:CharacterString>
  </individualName>
  <organisationName>
    <gco:CharacterString>
      <xsl:value-of select="INSTITUTE_NAME"/>
    </gco:CharacterString>
  </organisationName>
  <contactInfo>
    <CI_Contact>
      <xsl:if test="EMAIL">
        <address>
          <CI_Address>
            <electronicMailAddress>
              <gco:CharacterString>
                <xsl:value-of select="EMAIL"/>
              </gco:CharacterString>
            </electronicMailAddress>
          </CI_Address>
        </address>
      </xsl:if>
      <xsl:if test="INST_URL">
        <onlineResource>
          <CI_OnlineResource>
```



```
<linkage>
  <URL>
    <xsl:value-of select="INST_URL"/>
  </URL>
</linkage>
<function>
  <CI_OnLineFunctionCode
    codeList="{ $codelist }#CI_OnLineFunctionCode"

codeListValue="information">information</CI_OnLineFunctionCode>
  </function>
</CI_OnlineResource>
</onlineResource>
</xsl:if>
</CI_Contact>
</contactInfo>
</xsl:template>

<xsl:template name="listpointOfContact">
  <xsl:if test="$debug='showTplNames'">
    <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="'listpointOfContact'"/>
  </xsl:if>
  <xsl:for-each
select="/CERA2/CONTA/ROWSET/ROW[not (contains (CONTACT_TYPE, 'metadata'))]">
    <pointOfContact>
      <CI_ResponsibleParty>
        <xsl:call-template name="printResponsibleParty"/>
        <xsl:call-template name="printRole">
          <xsl:with-param name="role">
            <xsl:value-of select="CONTACT_TYPE"/>
          </xsl:with-param>
        </xsl:call-template>
      </CI_ResponsibleParty>
    </pointOfContact>
  </xsl:for-each>
</xsl:template>

<xsl:template name="printFormatRes">
  <xsl:if test="$debug='showTplNames'">
    <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="'printFormatRes'"/>
  </xsl:if>
  <xsl:for-each
select="/CERA2/FORMAT/ROWSET/ROW[not (contains (FORMAT_ACRONYM, 'pdf'))]">
    <resourceFormat>
      <MD_Format>
        <name>
          <xsl:variable name="formAC">
            <xsl:value-of select="FORMAT_ACRONYM"/>
          </xsl:variable>
          <gco:CharacterString>
            <xsl:choose>
```



```

        <xsl:when test="contains($formAC,'GRIB') or
contains($formAC,'GRIB')">
            <xsl:value-of select="'grb'"/>
        </xsl:when>
        <xsl:when test="contains($formAC,'netcdf') or
contains($formAC,'netCDF')">
            <xsl:value-of select="'nc'"/>
        </xsl:when>
        <xsl:when test="contains($formAC,'tar-') or
contains($formAC,'tar ')">
            <xsl:value-of select="'tar'"/>
        </xsl:when>
        <xsl:otherwise><xsl:value-of select="$formAC"/></xsl:otherwise>
    </xsl:choose>
    </gco:CharacterString>
</name>
<version>
    <gco:CharacterString>
        <xsl:value-of select="FORMAT_DESCR"/>
    </gco:CharacterString>
</version>
</MD_Format>
</resourceFormat>
</xsl:for-each>
</xsl:template>

```

```

<xsl:template name="printFormatDist">
    <xsl:for-each select="/CERA2/FORMAT/ROWSET/ROW[not
(contains(FORMAT_ACRONYM,'pdf') or contains(FORMAT_ACRONYM,'netcdf') or
contains(FORMAT_ACRONYM,'netCDF'))]">
        <xsl:if test="$debug='showTplNames'">
            <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="'printFormatDist'"/>
        </xsl:if>
        <distributionFormat>
            <MD_Format>
                <name>
                    <xsl:variable name="formAC">
                        <xsl:value-of select="FORMAT_ACRONYM"/>
                    </xsl:variable>
                    <gco:CharacterString>
                        <xsl:choose>
                            <xsl:when test="contains($formAC,'GRIB') or
contains($formAC,'GRIB')">
                                <xsl:value-of select="'grb'"/>
                            </xsl:when>
                            <xsl:when test="contains($formAC,'netcdf') or
contains($formAC,'netCDF')">
                                <xsl:value-of select="'nc'"/>
                            </xsl:when>
                            <xsl:when test="contains($formAC,'tar-') or
contains($formAC,'tar ')">
                                <xsl:value-of select="'tar'"/>
                            </xsl:when>

```



```

        <xsl:otherwise><xsl:value-of select="$formAC"/></xsl:otherwise>
    </xsl:choose>
</gco:CharacterString>
</name>
<version>
    <gco:CharacterString>
        <xsl:value-of select="FORMAT_DESCR"/>
    </gco:CharacterString>
</version>
</MD_Format>
</distributionFormat>
</xsl:for-each>
<xsl:if test="$MPI-OM != 'MPI-OM'">
    <distributionFormat>
        <MD_Format>
            <name>
                <gco:CharacterString>nc</gco:CharacterString>
            </name>
            <version>
                <gco:CharacterString>network Common Data
Format</gco:CharacterString>
            </version>
        </MD_Format>
    </distributionFormat>
</xsl:if>
</xsl:template>

<xsl:template name="printFormatResData">
    <xsl:for-each select="/CERA2/FORMAT_DS/ROWSET/ROW">
        <resourceFormat>
            <MD_Format>
                <name>
                    <xsl:variable name="formAC">
                        <xsl:value-of select="FORMAT_ACRONYM"/>
                    </xsl:variable>
                    <gco:CharacterString>
                        <xsl:choose>
                            <xsl:when test="contains($formAC,'GRIB') or
contains($formAC,'GRIB')">
                                <xsl:value-of select="'grb'"/>
                            </xsl:when>
                            <xsl:when test="contains($formAC,'netcdf') or
contains($formAC,'netCDF')">
                                <xsl:value-of select="'nc'"/>
                            </xsl:when>
                            <xsl:when test="contains($formAC,'tar-') or
contains($formAC,'tar ')">
                                <xsl:value-of select="'tar'"/>
                            </xsl:when>
                            <xsl:otherwise><xsl:value-of select="$formAC"/></xsl:otherwise>
                        </xsl:choose>
                    </gco:CharacterString>
                </name>
                <version>

```



```
<gco:CharacterString>
  <xsl:value-of select="FORMAT_DESCR"/>
</gco:CharacterString>
</version>
</MD_Format>
</resourceFormat>
</xsl:for-each>
</xsl:template>

<xsl:template name="printFormatDistData">
  <xsl:if test="$debug='showTplNames'">
    <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="'printFormatDistData'"/>
  </xsl:if>

  <xsl:variable name="formList">
    <xsl:for-each select="/CERA2/FORMAT_DS/ROWSET/ROW">
      <xsl:value-of select="FORMAT_ACRONYM"/>
    <xsl:value-of select="', '"/>
    </xsl:for-each>
  </xsl:variable>

  <xsl:for-each select="/CERA2/FORMAT_DS/ROWSET/ROW">
    <distributionFormat>
      <MD_Format>
        <name>
          <xsl:variable name="formAC">
            <xsl:value-of select="FORMAT_ACRONYM"/>
          </xsl:variable>
          <gco:CharacterString>
            <xsl:choose>
              <xsl:when test="contains($formAC,'GRIB') or
contains($formAC,'GRIB')">
                <xsl:value-of select="'grb'"/>
              </xsl:when>
              <xsl:when test="contains($formAC,'netcdf') or
contains($formAC,'netCDF')">
                <xsl:value-of select="'nc'"/>
              </xsl:when>
              <xsl:when test="contains($formAC,'tar-') or
contains($formAC,'tar ')">
                <xsl:value-of select="'tar'"/>
              </xsl:when>
              <xsl:otherwise><xsl:value-of select="$formAC"/></xsl:otherwise>
            </xsl:choose>
          </gco:CharacterString>
        </name>
        <version>
          <gco:CharacterString>
            <xsl:value-of select="FORMAT_DESCR"/>
          </gco:CharacterString>
        </version>
      </MD_Format>
    </distributionFormat>
  </xsl:for-each>
</xsl:template>
```



```

</xsl:for-each>
</xsl:template>

<xsl:template name="identificationInfo">
  <xsl:if test="$debug='showTmplNames'">
    <xsl:value-of select="$debugInfoTmpl"/><xsl:value-of
select="'identificationInfo'" />
  </xsl:if>
  <xsl:variable name="id"><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_ID"/></xsl:variable>
  <identificationInfo>
    <MD_DataIdentification>
      <citation>
        <CI_Citation>
          <xsl:variable name="citation_complete">
            <xsl:choose>
              <xsl:when test="$doiSTR!=''">
                <xsl:for-each
select="/CERA2/REFER/ROWSET/ROW[(REF_TYPE_ID=$doi_exp_id) or
(REF_TYPE_ID=$doi_data_id)]">
                  <xsl:value-of
select="AUTHORS"/>
                  <xsl:value-of select="'
'"/>
                  <xsl:value-of
select="substring-before(PUBLICATION_DATE, '-')"/>
                  <xsl:value-of
select="'; '"/>
                  <xsl:value-of
select="$entryAcro"/>
                  <xsl:value-of
select="'. '"/>
                </xsl:for-each>
                </xsl:when>
                <xsl:otherwise>
                  <xsl:for-each
select="/CERA2/CITATION/ROWSET/ROW">
                    <xsl:value-of
select="NAME"/>
                    </xsl:for-each>
                    <xsl:value-of select="', '"/>
                    <xsl:value-of
select="substring-before(/CERA2/ENTRY/ROWSET/ROW/CREATION_DATE, '-')"/>
                    <xsl:value-of select="': '"/>
                    <xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_NAME"/>
                    <xsl:value-of select="'.
CERA-DB &#34;'"/>
                    <xsl:value-of
select="$entryAcro"/>
                    <xsl:value-of
select="'&#34;'"/>
                  </xsl:otherwise>

```



```

        </xsl:choose>
    </xsl:variable>
    <xsl:variable name="citation">
        <xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_NAME"/>
    </xsl:variable>
    <title>
        <gco:CharacterString>
            <xsl:value-of select="$citation"/>
        </gco:CharacterString>
    </title>
    <date>
        <CI_Date>
            <date>
                <gco:Date>
                    <xsl:value-of select="substring-
before (/CERA2/ENTRY/ROWSET/ROW/CREATION_DATE,'T')"/>
                </gco:Date>
            </date>
        </dateType>
        <CI_DateTypeCode
codeList="{ $codelist }#CI_DateTypeCode"
codeListValue="creation">creation</CI_DateTypeCode>
    </dateType>
    </CI_Date>
</date>
        <identifier>
            <MD_Identifier>
                <code>
                    <xsl:call-template
name="dataSetURI"/>
                </code>
            </MD_Identifier>
        </identifier>
        <xsl:call-template name="printcitedResponsibleParty"/>
    </CI_Citation>
</citation>

    <abstract>
        <gco:CharacterString><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/SUMMARY"/></gco:CharacterString>
    </abstract>

    <xsl:call-template name="listpointOfContact"/>

    <xsl:call-template name="printFormatRes"/>
    <descriptiveKeywords>
        <MD_Keywords>
            <xsl:for-each select="/CERA2/KEY_C/ROWSET/ROW">
                <keyword>
                    <gco:CharacterString>
                        <xsl:value-of select="GENERAL_KEY"/>
                    </gco:CharacterString>
                </keyword>
            </xsl:for-each>
        </MD_Keywords>
    </descriptiveKeywords>

```



```
<type>
  <MD_KeywordTypeCode codeList="{ $codelist}#MD_KeywordTypeCode"
codeListValue="theme" >theme</MD_KeywordTypeCode>
</type>
</MD_Keywords>
</descriptiveKeywords>

<resourceConstraints>
  <MD_LegalConstraints>
    <xsl:for-each select="/CERA2/DISTR_ALL/ROWSET/ROW">
      <xsl:variable name="access"><xsl:value-of
select="ACCESS_CONSTRAINT_DESCR"/></xsl:variable>
      <xsl:variable name="accessVal">
        <xsl:choose>
          <xsl:when test="contains($access,':')">
            <xsl:value-of select="normalize-
space(substring-after($access,':'))"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of
select="$access"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
      <xsl:variable name="accessCode">
        <xsl:choose>
          <xsl:when test="contains($access,':')">
            <xsl:value-of select="normalize-
space(substring-before($access,':'))"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of
select="$access"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
      <xsl:if test="($accessVal!='') and ($accessVal!='unrestricted')
and ($accessVal!='not filled') ">
        <xsl:variable name="access_code">
          <xsl:choose>
            <xsl:when
test="contains($accessCode,'copyright')">
              <xsl:value-
of select="'copyright'"/>
            </xsl:when>
            <xsl:when test="$accessCode =
'restricted'">
              <xsl:value-
of select="'restricted'"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-
of select="'otherRestrictions'"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
      </xsl:if>
    </xsl:for-each>
  </MD_LegalConstraints>
</resourceConstraints>
```



```
        </xsl:variable>
    <accessConstraints>
        <MD_RestrictionCode
codeList="{ $codelist }#MD_RestrictionCode"
            codeListValue="{ $access_code }">
            <xsl:choose>
                <xsl:when
test="contains($accessVal, 'not available')">
                    <xsl:value-of select="$C3Grid_NO_ACCESS"/>
                </xsl:when>
                <xsl:otherwise>

    <xsl:value-of select="$accessVal"/>

                </xsl:otherwise>
            </xsl:choose>
        </MD_RestrictionCode>
    </accessConstraints>
</xsl:if>
</xsl:for-each>
    <xsl:for-each select="/CERA2/DISTR_ALL/ROWSET/ROW">
        <xsl:variable name="use"><xsl:value-of
select="USE_CONSTRAINT_DESCR"/></xsl:variable>
        <xsl:variable name="useVal">
            <xsl:choose>
                <xsl:when test="contains($use, ':')">
                    <xsl:value-of select="$use"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of
select="$use"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:variable name="useCode">
            <xsl:choose>
                <xsl:when test="contains($use, ':')">
                    <xsl:value-of select="normalize-
space(substring-before($use, ':'))"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of
select="$use"/>
                </xsl:otherwise>
            </xsl:choose>
        </xsl:variable>
        <xsl:if test="($useVal!='') and ($useVal!='unrestricted') and
($useVal!='not filled') ">
            <xsl:variable name="use_code">
                <xsl:choose>
                    <xsl:when
test="contains($useCode, 'copyright')">
                        <xsl:value-
of select="'copyright'"/>
                    </xsl:when>
```



```

'restricted'")
of select="'restricted'"/>
of select="'otherRestrictions'"/>
</xsl:variable>
<useConstraints>
  <MD_RestrictionCode
codeList="{ $codelist }#MD_RestrictionCode"
codeListValue="{ $use_code }">
  <xsl:choose>
    <xsl:when
test="contains($useVal, 'not available')">
      <xsl:value-of select="$C3Grid_NO_ACCESS"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$useVal"/>
    </xsl:otherwise>
  </xsl:choose>
  </MD_RestrictionCode>
</useConstraints>
</xsl:if>
</xsl:for-each>
<!--
  <otherConstraints>
    <gco:CharacterString>
      AAAAAA<xsl:value-of
select="/CERA2/CHECK_EXP_Permission/ROWSET/ROW/permission"/>BBBBBB
    <xsl:choose>
      <xsl:when test="/CERA2/CHECK_EXP_PERM/ROWSET/ROW/PERM=1">
        <xsl:value-of select="'C3GRID_BASIC_ACCESS=FALSE'"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="'C3GRID_BASIC_ACCESS=TRUE'"/>
      </xsl:otherwise>
    </xsl:choose>
  </gco:CharacterString>
</otherConstraints>
-->
  </MD_LegalConstraints>
</resourceConstraints>
<xsl:for-each
select="/CERA2/TOPIC_ALL/ROWSET/ROW[contains(REFERENCE_METHOD, 'non
gridded')=false()]">
  <xsl:if test="(contains(REFERENCE_METHOD, 'gridded')=true()) and
(position() = 1)">
    <spatialRepresentationType>
      <MD_SpatialRepresentationTypeCode

```




```
</gco:Decimal>
</westBoundLongitude>
<eastBoundLongitude>
  <gco:Decimal>
    <xsl:value-of select="MAX_LON"/>
  </gco:Decimal>
</eastBoundLongitude>
<southBoundLatitude>
  <gco:Decimal>
    <xsl:value-of select="MIN_LAT"/>
  </gco:Decimal>
</southBoundLatitude>
<northBoundLatitude>
  <gco:Decimal>
    <xsl:value-of select="MAX_LAT"/>
  </gco:Decimal>
</northBoundLatitude>
</EX_GeographicBoundingBox>
</geographicElement>
  </xsl:if>
</xsl:for-each>

  <xsl:for-each select="/CERA2/COVER/ROWSET/ROW">
    <xsl:if test="position()=1">
      <xsl:variable name="timeShift2">
        <xsl:value-of
select="CURRENTNESS_REF_DESCR"/>
      </xsl:variable>
      <xsl:variable name="timeShiftStart">
        <xsl:if test="contains($timeShift2,'time before present')">
          <xsl:if test="contains($timeShift2,'(')">
            <xsl:value-of
select="substring-before(substring-before(substring-
after($timeShift2,'('),')'),'=)"/>
          </xsl:if>
        </xsl:if>
      </xsl:variable>
      <xsl:variable name="timeShiftDiff">
        <xsl:if test="contains($timeShift2,'time before present')">
          <xsl:if test="contains($timeShift2,'(')">
            <xsl:value-of
select="substring-after(substring-before(substring-
after($timeShift2,'('),')'),'=)"/>
          </xsl:if>
        </xsl:if>
      </xsl:variable>
      <xsl:variable name="tmBegin2">
        <xsl:choose>
          <xsl:when test="(START_YEAR &#60; 0) and
(string-length(string(START_YEAR))=2)">
            <xsl:value-of select="'000'"/>
          </xsl:when>
          <xsl:when test="(START_YEAR &#60; 0) and
(string-length(string(START_YEAR))=3)">
```



```

                                <xsl:value-of select="'00'"/>
                                </xsl:when>
                                <xsl:when test="(START_YEAR &#60; 0) and
(string-length(string(START_YEAR))=4)">
                                <xsl:value-of select="'0'"/>
                                </xsl:when>
                                <xsl:when test="(START_YEAR &#62; 0) and
(string-length(string(START_YEAR))=1)">
                                <xsl:value-of select="'000'"/>
                                </xsl:when>
                                <xsl:when test="(START_YEAR &#62; 0) and
(string-length(string(START_YEAR))=2)">
                                <xsl:value-of select="'00'"/>
                                </xsl:when>
                                <xsl:when test="(START_YEAR &#62; 0) and
(string-length(string(START_YEAR))=3)">
                                <xsl:value-of select="'0'"/>
                                </xsl:when>
                                <xsl:otherwise>
                                <xsl:value-of select="''"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                <!--
                                <xsl:value-of select="START_YEAR + $timeShiftStart +
$timeShiftDiff"/>
                                -->
                                <xsl:value-of select="START_YEAR"/>
                                <xsl:choose>
                                <xsl:when test="string-
length(string(START_MONTH))=1">
                                <xsl:value-of select="'-0'"/>
                                </xsl:when>
                                <xsl:otherwise>
                                <xsl:value-of select="'-'"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                <xsl:value-of select="START_MONTH"/>
                                <xsl:choose>
                                <xsl:when test="string-
length(string(START_DAY))=1">
                                <xsl:value-of select="'-0'"/>
                                </xsl:when>
                                <xsl:otherwise>
                                <xsl:value-of select="'-'"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                <xsl:value-of select="START_DAY"/>
                                <xsl:value-of select="'T00:00:00'"/>
                                </xsl:variable>
                                <xsl:variable name="tmBegin">
                                <xsl:value-of select="$tmBegin2" />
                                <!--
                                <xsl:call-template name="date:add">
                                <xsl:with-param name="date-time"
select="$tmBegin2" />

```



```

select="'PT0H'" />
                                <xsl:with-param name="duration"
                                </xsl:call-template>
                                -->
                                </xsl:variable>
                                <xsl:variable name="tmEnd1">
                                <xsl:choose>
                                <xsl:when test="(STOP_YEAR &#60; 0) and
(string-length(string(STOP_YEAR))=2)">
                                <xsl:value-of select="'000'"/>
                                </xsl:when>
                                <xsl:when test="(STOP_YEAR &#60; 0) and
(string-length(string(STOP_YEAR))=3)">
                                <xsl:value-of select="'00'"/>
                                </xsl:when>
                                <xsl:when test="(STOP_YEAR &#60; 0) and
(string-length(string(STOP_YEAR))=4)">
                                <xsl:value-of select="'0'"/>
                                </xsl:when>
                                <xsl:when test="(STOP_YEAR &#62; 0) and
(string-length(string(STOP_YEAR))=1)">
                                <xsl:value-of select="'000'"/>
                                </xsl:when>
                                <xsl:when test="(STOP_YEAR &#62; 0) and
(string-length(string(STOP_YEAR))=2)">
                                <xsl:value-of select="'00'"/>
                                </xsl:when>
                                <xsl:when test="(STOP_YEAR &#62; 0) and
(string-length(string(STOP_YEAR))=3)">
                                <xsl:value-of select="'0'"/>
                                </xsl:when>
                                <xsl:otherwise>
                                <xsl:value-of select="''"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                <!--
                                <xsl:value-of select="STOP_YEAR + $timeShiftStart +
$timeShiftDiff"/>
                                -->
                                <xsl:value-of select="STOP_YEAR"/>
                                <xsl:choose>
                                <xsl:when test="string-
length(string(STOP_MONTH))=1">
                                <xsl:value-of select="'-0'"/>
                                </xsl:when>
                                <xsl:otherwise>
                                <xsl:value-of select="'-'"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                <xsl:value-of select="STOP_MONTH"/>
                                <xsl:choose>
                                <xsl:when test="string-
length(string(STOP_DAY))=1">
                                <xsl:value-of select="'-0'"/>

```



```

        </xsl:when>
        <xsl:otherwise>
        <xsl:value-of select="'-'"/>
        </xsl:otherwise>
    </xsl:choose>
    <xsl:value-of select="STOP_DAY"/>
    <xsl:value-of select="'T00:00:00'"/>
        </xsl:variable>
    <xsl:variable name="tmEnd2">
    <xsl:choose>
    <xsl:when test="contains($tmEnd1,'T00:00')">
        <xsl:call-template name="date:add">
            <xsl:with-param name="date-time"
select="$tmEnd1" />
            <xsl:with-param name="duration"
select="'P1D'" />
        </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of
select="$tmEnd1"/>
        </xsl:otherwise>
    </xsl:choose>
    </xsl:variable>
    <xsl:variable name="tmEnd">
    <xsl:choose>
    <xsl:when test="contains($tmEnd1,'T00:00')">
        <xsl:call-template name="date:add">
            <xsl:with-param name="date-time"
select="$tmEnd2" />
            <xsl:with-param name="duration"
select="'-PT1H'" />
        </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of
select="$tmEnd2"/>
        </xsl:otherwise>
    </xsl:choose>
    </xsl:variable>

        <xsl:variable
name="valBegin"><xsl:value-of select="translate($tmBegin,'
','')"/></xsl:variable>
        <xsl:variable
name="valEnd"><xsl:value-of select="translate($tmEnd,' ','')"/></xsl:variable>
        <xsl:variable name="tmStepCnt">
            <xsl:choose>
            <xsl:when
test="/CERA2/SUM_STEP_COUNT/ROWSET/ROW/TOT_STEP_COUNT!=0">
                <xsl:value-
of select="/CERA2/SUM_STEP_COUNT/ROWSET/ROW/TOT_STEP_COUNT"/>
            </xsl:when>
            <xsl:otherwise>0</xsl:otherwise>

```



```

</xsl:choose>
</xsl:variable>
<xsl:variable name="stepStepCnt">
  <xsl:value-of
select="/CERA2/EXT_STEP/ROWSET/ROW/TARGET_STEP_COUNT"/>
</xsl:variable>
<!--
  AA<xsl:value-of select="$tmStepCnt"/>AA
-->
<xsl:if test="$tmStepCnt!=0">
  <xsl:variable
name="tmStep1">
    <xsl:choose>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_YEAR!=0"><xsl:value-of
select="/CERA2/EXT_STEP/ROWSET/ROW/STEP_YEAR"/></xsl:when>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_MONTH!=0"><xsl:value-of
select="/CERA2/EXT_STEP/ROWSET/ROW/STEP_MONTH"/></xsl:when>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_DAY!=0"><xsl:value-of
select="/CERA2/EXT_STEP/ROWSET/ROW/STEP_DAY"/></xsl:when>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_HOUR!=0"><xsl:value-of
select="/CERA2/EXT_STEP/ROWSET/ROW/STEP_HOUR"/></xsl:when>
      <xsl:otherwise>0</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable name="tmUnit">
    <xsl:choose>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_YEAR!=0">year</xsl:when>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_MONTH!=0">month</xsl:when>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_DAY!=0">day</xsl:when>
      <xsl:when
test="/CERA2/EXT_STEP/ROWSET/ROW/STEP_HOUR!=0">hour</xsl:when>
      <xsl:otherwise><xsl:value-of
select="$MyDummy"/></xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable
name="durTotVal">
    <xsl:choose>
      <xsl:when
test="floor($tmStep1 div $stepStepCnt)=1">
        <xsl:value-of select="'1#'" /><xsl:value-of select="$tmUnit" />
      </xsl:when>
      <xsl:when
test="floor($tmStep1 div $stepStepCnt)=0">

```



```

        <xsl:call-template name="calcStepData">
        <xsl:with-param name="start" select="$tmBegin"/>
        <xsl:with-param name="end" select="$tmEnd"/>
            <xsl:with-param name="tmStepCnt"
                <xsl:with-param name="stepStepCnt"
                    <xsl:with-param name="tmStepP"
                        <xsl:with-param name="tmUnit"
                            </xsl:call-template>
                                </xsl:when>
                                    </xsl:choose>
                                        </xsl:variable>
                                            <xsl:variable
                                                name="valStep">
                                                    <xsl:value-of
                                                        select="translate(substring-before($durTotVal,'#'),' ','')"/>
                                                    </xsl:variable>
                                                        <xsl:variable
                                                            name="valUnit">
                                                                <xsl:value-of
                                                                    select="substring-after($durTotVal,'#')"/>
                                                                </xsl:variable>
                                                                    <xsl:variable
                                                                        name="valStepCnt">
                                                                            <xsl:value-of select="$tmStepCnt"/>
                                                                            </xsl:variable>
                                                                                <temporalElement>
                                                                                    <EX_TemporalExtent>
                                                                                        <extent>
                                                                                            <gml:TimePeriod
                                                                                                gml:id="extent_{$valStepCnt}">
                                                                                                    <gml:beginPosition><xsl:value-of
                                                                                                        select="$valBegin"/></gml:beginPosition>
                                                                                                        <gml:endPosition><xsl:value-of
                                                                                                            select="$valEnd"/></gml:endPosition>
                                                                                                                <gml:timeInterval unit="{ $valUnit}" factor="-1"
                                                                                                                    radix="{ $valStep}"><xsl:value-of select="$valStepCnt"/></gml:timeInterval>
                                                                                                                        </gml:TimePeriod>
                                                                                                                            </extent>
                                                                                                                                </EX_TemporalExtent>
                                                                                                                                    </temporalElement>
                                                                                                                                        </xsl:if>
                                                                                                                                            <xsl:if test="$tmStepCnt=0">
                                                                                                                                                <xsl:for-each
                                                                                                                                                    select="/CERA2/MOMENT/ROWSET/ROW">
                                                                                                                                                        <xsl:variable
                                                                                                                                                            name="tmUnit">
                                                                                                                                            <xsl:choose>

```



```

        <xsl:when test="STEP_YEAR!=0">year</xsl:when>
        <xsl:when
test="STEP_MONTH!=0">month</xsl:when>
        <xsl:when test="STEP_DAY!=0">day</xsl:when>
        <xsl:when test="STEP_HOUR!=0">hour</xsl:when>
        <xsl:otherwise><xsl:value-of
select="$MyDummy"/></xsl:otherwise>
        </xsl:choose>
        </xsl:variable>
        <xsl:variable
name="valUnit">
        <xsl:value-of
select="$tmUnit"/>
        </xsl:variable>
        <xsl:variable
name="tmStep">
        <xsl:choose>
        <xsl:when test="STEP_YEAR!=0"><xsl:value-of
select="STEP_YEAR"/></xsl:when>
        <xsl:when
test="STEP_MONTH!=0"><xsl:value-of select="STEP_MONTH"/></xsl:when>
        <xsl:when test="STEP_DAY!=0"><xsl:value-of
select="STEP_DAY"/></xsl:when>
        <xsl:when test="STEP_HOUR!=0"><xsl:value-of
select="STEP_HOUR"/></xsl:when>
        <xsl:otherwise>0</xsl:otherwise>
        </xsl:choose>
        </xsl:variable>
        <xsl:variable
name="valStepCnt">
        <xsl:value-of
select="NUM_POINTS"/>
        </xsl:variable>
        <!--
        BB<xsl:value-of select="$valStepCnt"/>BB
        -->
        <xsl:variable
name="valStepCnt2">
        <xsl:call-template name="calcStepCnt">
        <xsl:with-param name="start" select="$tmBegin"/>
        <xsl:with-param name="end" select="$tmEnd"/>
        <xsl:with-param name="tmStepDur" select="$tmStep"/>
        </xsl:call-template>
        </xsl:variable>
        <xsl:variable
name="valStep">
        <xsl:value-of
select="$tmStep"/>
        </xsl:variable>
        <temporalElement>
        <EX_TemporalExtent>
        <extent>
        <gml:TimePeriod
gml:id="extent_{$valStepCnt}">

```



```

select="$valBegin"/></gml:beginPosition>
select="$valEnd"/></gml:endPosition>

    <gml:timeInterval unit="{ $valUnit}" factor="-1"
radix="{ $valStep}"><xsl:value-of select="$valStepCnt"/></gml:timeInterval>
        </gml:TimePeriod>
    </extent>
</EX_TemporalExtent>
</temporalElement>
</xsl:for-each>
</xsl:if>

    </xsl:if>
</xsl:for-each>
<xsl:variable name="ceraProj">
    <xsl:value-of select="$projAcro"/>
</xsl:variable>
<!--
    AA<xsl:value-of select="contains($dataType, 'DKRZ')"/>AA
    AA<xsl:value-of select="contains($expSummary, 'OPA')"/>AA
    AA<xsl:value-of select="contains($expSummary, 'NEMO')"/>AA
-->
        <xsl:choose>
            <xsl:when test="contains($entryAcro, 'OM-
GR1.5_EH5-T63') or contains($entryAcro, 'OM-GR1.5L40_EH5-T63')">
<!--
                <xsl:value-of select="'AAAA'"/>
-->
                    <xsl:for-each
select="/CERA2/COVER/ROWSET/ROW">
                        <xsl:variable
name="vertC3CRSunit">
                            <xsl:choose>
                                <xsl:when
test="MIN_ALT_UNIT_ACRONYM='m' or MAX_ALT_UNIT_ACRONYM='m'">
                                    <xsl:value-of select="'m'"/>
                                </xsl:when>
                                <xsl:when
test="MIN_ALT_UNIT_ACRONYM='hPa' and MAX_ALT_UNIT_ACRONYM='hPa'">
                                    <xsl:value-of select="'hPa'"/>
                                </xsl:when>
                                <xsl:when
test="MIN_ALT_UNIT_ACRONYM='model level no.' or MAX_ALT_UNIT_ACRONYM='model
level no.'">
                                    <xsl:value-of
select="'hybridModelLevel'"/>
                                </xsl:when>
                                <xsl:otherwise>
                                    <xsl:value-of select="'m'"/>
                                </xsl:otherwise>
                            </xsl:choose>
                        </xsl:variable>
                    <verticalElement>

```



```

        <EX_VerticalExtent
id="{ $verticalElementPrefix } { $vertC3CRSunit } ">
        <minimumValue>
            <gco:Real>
                <xsl:value-of
select="MIN_ALTITUDE" />
            </gco:Real>
        </minimumValue>
        <maximumValue>
            <gco:Real>
                <xsl:value-of select="MAX_ALTITUDE" />
            </gco:Real>
        </maximumValue>

name="vertC3CRS">
                                <xsl:variable
                                <xsl:choose>
                                    <xsl:when
test="MIN_ALT_UNIT_ACRONYM='m' or MAX_ALT_UNIT_ACRONYM='m'">
                                        <xsl:value-of
select="'vertCRS.depthBeneathSurface'"/>
                                    </xsl:when>
                                    <xsl:when
test="MIN_ALT_UNIT_ACRONYM='hPa' and MAX_ALT_UNIT_ACRONYM='hPa'">
                                        <xsl:value-of
select="'vertCRS.standardPressureLevel'"/>
                                    </xsl:when>
                                    <xsl:when
test="MIN_ALT_UNIT_ACRONYM='model level no.' and MAX_ALT_UNIT_ACRONYM='model
level no.'">
                                        <xsl:value-of
select="'vertCRS.hybridModelLevelL'"/>
                                    </xsl:when>
                                <xsl:otherwise>
                                    <xsl:value-of
select="'vertCRS.depthBeneathSurface'"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                </xsl:variable>
                                <xsl:variable
name="topicList">
                                    <xsl:value-of select="'add_info,'"/>
                                </xsl:variable>
                                <verticalCRS
                                <xlink:href="{ $verticalCRSTemplateURL } #xpointer ( // * [ @gml : id = ' { $vertC3CRS } ' ] ) "
                                <xlink:title="CRS for Height"
                                <xlink:role="{ substring-
before ( concat ( $topicList , '# ' ) , ', # ' ) }"/>
                                </EX_VerticalExtent>
                            </verticalElement>
                        </xsl:for-each>
                    </xsl:when>

```



```
<!-- Flat File projects contains 'DKRZ' but not ocean data -->
<xsl:when test="contains($dataType,'DKRZ') and not
(contains($expSummary,'OPA') or contains($expSummary,'NEMO'))">
<!--
  <xsl:value-of select="'BBBB'"/>
  -->
  <xsl:for-each select="/CERA2/TOPIC_ALL_VERT_SC/ROWSET/ROW">
    <xsl:variable name="level">
      <xsl:value-of select="VERT_SC"/>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="number($level) &#62; 18">
        <verticalElement>
          <EX_VericalExtent
id="{ $verticalElementPrefix}L{$level}">
            <minimumValue>
              <gco:Real>
                <xsl:value-of select="'1'"/>
              </gco:Real>
            </minimumValue>
            <maximumValue>
              <gco:Real>
                <xsl:value-of select="$level"/>
              </gco:Real>
            </maximumValue>
            <xsl:variable name="vertC3CRS">
              <xsl:value-of
select="concat('vertCRS.hybridModelLevelL', $level)"/>
            </xsl:variable>
            <xsl:variable name="topicList">
              <xsl:value-of select="'add_info,'"/>
            </xsl:variable>
            <verticalCRS
xlink:href="{ $verticalCRSTemplateURL}#xpointer(//*[@gml:id='{ $vertC3CRS}'])"
              xlink:title="CRS for Height"
              xlink:role="{substring-
before(concat($topicList,'#'),',#')}">
            </EX_VericalExtent>
          </verticalElement>
          <verticalElement>
            <EX_VericalExtent
id="{ $verticalElementPrefix}hPa">
              <minimumValue>
                <gco:Real>
                  <xsl:value-of select="'10'"/>
                </gco:Real>
              </minimumValue>
              <maximumValue>
                <gco:Real>
                  <xsl:value-of select="'1000'"/>
                </gco:Real>
              </maximumValue>
            <xsl:variable name="vertC3CRS">
```




```

MAXIMUM_ALT">
    <xsl:if test="MINIMUM_ALT or
    <verticalElement>
      <EX_VerticalExtent
id="{verticalElementPrefix}hPa">
    <minimumValue>
      <gco:Real>
        <xsl:value-of select="MINIMUM_ALT"/>
      </gco:Real>
    </minimumValue>
    <maximumValue>
      <gco:Real>
        <xsl:value-of select="MAXIMUM_ALT"/>
      </gco:Real>
    </maximumValue>
    <xsl:variable
name="vertC3CRS">
      <xsl:value-of
select="'vertCRS.standardPressureLevel'"/>
    </xsl:variable>
    <xsl:variable
name="topicList">
      <xsl:value-of select="'add_info,'"/>
    </xsl:variable>
    <verticalCRS
xlink:href="{verticalCRSTemplateURL}#xpointer(//*[@gml:id='{verticalC3CRS}'])"
      xlink:title="CRS for Height"
      xlink:role="{substring-
before(concat($topicList,'#'),',#')}">
    </EX_VerticalExtent>
  </verticalElement>
</xsl:if>
</xsl:for-each>
<xsl:if test="contains($dataType,'CERA') and position()=1">
<!--
  <xsl:value-of select="'CCCC-111'"/>
  -->
  <xsl:for-each select="/CERA2/COVER_ALL_m/ROWSET/ROW">
    <verticalElement>
      <EX_VerticalExtent id="{verticalElementPrefix}m">
        <minimumValue>
          <gco:Real>
            <xsl:value-of select="'0'"/>
          </gco:Real>
        </minimumValue>
        <maximumValue>
          <gco:Real>
            <xsl:value-of select="'0'"/>
          </gco:Real>
        </maximumValue>
        <xsl:variable name="vertC3CRS">
          <xsl:value-of
select="'vertCRS.heightAboveSurface'"/>

```



```

</xsl:variable>
<xsl:variable name="topicList">
  <xsl:value-of select="'add_info,'"/>
</xsl:variable>
<verticalCRS

xlink:href="{ $verticalCRSTemplateURL}#xpointer(//*[@gml:id='{ $vertC3CRS}'])"
  xlink:title="CRS for Height"
  xlink:role="{substring-
before(concat($topicList,'#'),',#')}" />
  </EX_VerticalExtent>
</verticalElement>
</xsl:for-each>
</xsl:if>
<!-- Ocean data and HOAPS projects -->
  <xsl:if test="(contains($dataType,'DKRZ') and
(contains($expSummary,'OPA') or contains($expSummary,'NEMO')))">
<!--
    <xsl:value-of select="'DKRZ/NEMO'"/>
    <xsl:value-of select="'CCCC-222'"/>
  -->
    <xsl:for-each
select="/CERA2/COVER_ALL_m/ROWSET/ROW">
      <xsl:if test="MINIMUM_ALT or
MAXIMUM_ALT">
        </xsl:if>
        <verticalElement>
          <EX_VerticalExtent
id="{ $verticalElementPrefix}m">
            <minimumValue>
              <gco:Real>
                <xsl:value-of
select="MINIMUM_ALT"/>
              </gco:Real>
            </minimumValue>
            <maximumValue>
              <gco:Real>
                <xsl:value-of
select="MAXIMUM_ALT"/>
              </gco:Real>
            </maximumValue>
            <xsl:variable
name="vertC3CRS">
              <xsl:value-of
select="'vertCRS.depthBeneathSurface'"/>
            </xsl:variable>
            <xsl:variable
name="topicList">
              <xsl:value-of select="'add_info,'"/>
            </xsl:variable>
          </verticalCRS

```




```

"/>
select="substring-before(PUBLICATION_DATE, '-')"/>
select="'; "/>
select="$entryAcro"/>
select="'. "/>
</xsl:for-each>
</xsl:when>
<!-- this experiment has no DOI -->
<xsl:otherwise>
  <xsl:for-each
select="/CERA2/CITATION/ROWSET/ROW[position()=1]"/>
  <xsl:value-of
select="NAME"/>
  </xsl:for-each>
  <xsl:value-of select="', "/>
  <xsl:value-of
select="substring-before(/CERA2/ENTRY/ROWSET/ROW/CREATION_DATE, '-')"/>
  <xsl:value-of select="': "/>
  <xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_NAME"/>
  <xsl:value-of select="'.
CERA-DB &#34;"/>
  <xsl:value-of
select="$entryAcro"/>
  <xsl:value-of select="'&#34;
"/>
  </xsl:otherwise>
</xsl:choose>
</xsl:variable>
<title>
  <gco:CharacterString>
    <xsl:value-of select="$citation"/>
  </gco:CharacterString>
</title>
<date>
  <CI_Date>
    <date>
      <gco:Date>
        <xsl:value-of select="substring-
before(/CERA2/ENTRY/ROWSET/ROW/CREATION_DATE, ' ')/>
      </gco:Date>
    </date>
  <dateType>
    <CI_DateTypeCode
codeList="{ $codelist }#CI_DateTypeCode"
codeListValue="creation">creation</CI_DateTypeCode>
  </dateType>
</CI_Date>
</date>
  <identifier>

```



```
<MD_Identifier>
  <code>
    <xsl:call-template
name="dataSetURI"/>
    </code>
  </MD_Identifier>
</identifier>
<xsl:call-template name="printcitedResponsibleParty"/>
</CI_Citation>
</citation>

<abstract>
  <gco:CharacterString><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/SUMMARY"/></gco:CharacterString>
</abstract>
<!-- actual not used
<status>
  <xsl:choose>
    <xsl:when
test="/CERA2/ENTRY/ROWSET/ROW/PROGRESS_ACRONYM='complete'">
      <MD_ProgressCode codeList="{ $codelist }#MD_ProgressCode"
codeListValue="completed">completed</MD_ProgressCode>
    </xsl:when>
    <xsl:when
test="/CERA2/ENTRY/ROWSET/ROW/PROGRESS_ACRONYM='complete, processed'">
      <MD_ProgressCode codeList="{ $codelist }#MD_ProgressCode"
codeListValue="completed">completed</MD_ProgressCode>
    </xsl:when>
    <xsl:when
test="/CERA2/ENTRY/ROWSET/ROW/PROGRESS_ACRONYM='complete, original'">
      <MD_ProgressCode codeList="{ $codelist }#MD_ProgressCode"
codeListValue="completed">completed</MD_ProgressCode>
    </xsl:when>
    <xsl:when test="/CERA2/ENTRY/ROWSET/ROW/PROGRESS_ACRONYM='in
work'">
      <MD_ProgressCode codeList="{ $codelist }#MD_ProgressCode"
codeListValue="underDevelopment">underDevelopment</MD_ProgressCode>
    </xsl:when>
    <xsl:when
test="/CERA2/ENTRY/ROWSET/ROW/PROGRESS_ACRONYM='planned/ordered'">
      <MD_ProgressCode codeList="{ $codelist }#MD_ProgressCode"
codeListValue="planned">planned</MD_ProgressCode>
    </xsl:when>
  </xsl:choose>
</status>
-->
<xsl:call-template name="listpointOfContact"/>
<!--
-->
<xsl:call-template name="printFormatResData"/>
<resourceConstraints>
  <MD_LegalConstraints>
    <xsl:for-each select="/CERA2/DISTR/ROWSET/ROW">
      <xsl:variable name="access"><xsl:value-of
select="ACCESS_CONSTRAINT_DESCR"/></xsl:variable>
```



```
<xsl:variable name="accessVal">
  <xsl:choose>
    <xsl:when test="contains($access,':')">
      <xsl:value-of select="normalize-
space(substring-after($access,':'))"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of
select="$access"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
<xsl:variable name="accessCode">
  <xsl:choose>
    <xsl:when test="contains($access,':')">
      <xsl:value-of select="normalize-
space(substring-before($access,':'))"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of
select="$access"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
<xsl:if test="($accessVal!='') and ($accessVal!='unrestricted')
and ($accessVal!='not filled') ">
  <xsl:variable name="access_code">
    <xsl:choose>
      <xsl:when
test="contains($accessCode,'copyright')">
        <xsl:value-
of select="'copyright'"/>
      </xsl:when>
      <xsl:when test="$accessCode =
'restricted'">
        <xsl:value-
of select="'restricted'"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-
of select="'otherRestrictions'"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <accessConstraints>
    <MD_RestrictionCode
codeList="{ $codelist }#MD_RestrictionCode"
codeListValue="{ $access_code }">
    <xsl:choose>
      <xsl:when
test="contains($accessVal,'not available')">
      <xsl:value-of select="$C3Grid_NO_ACCESS"/>
    </xsl:when>
    <xsl:otherwise>
```



```
<xsl:value-of select="$accessVal"/>
                                </xsl:otherwise>
                                </xsl:choose>
                                </MD_RestrictionCode>
                                </accessConstraints>
                                </xsl:if>
                                </xsl:for-each>

                                <xsl:for-each select="/CERA2/DISTR/ROWSET/ROW">
                                    <xsl:variable name="use"><xsl:value-of
select="USE_CONSTRAINT_DESCR"/></xsl:variable>
                                    <xsl:variable name="useVal">
                                        <xsl:choose>
                                            <xsl:when test="contains($use,':')">
                                                <xsl:value-of select="normalize-
space(substring-after($use,':'))"/>
                                                </xsl:when>
                                                <xsl:otherwise>
                                                    <xsl:value-of
select="$use"/>
                                                    </xsl:otherwise>
                                                </xsl:choose>
                                            </xsl:variable>
                                            <xsl:variable name="useCode">
                                                <xsl:choose>
                                                    <xsl:when test="contains($use,':')">
                                                        <xsl:value-of select="normalize-
space(substring-before($use,':'))"/>
                                                        </xsl:when>
                                                        <xsl:otherwise>
                                                            <xsl:value-of
select="$use"/>
                                                            </xsl:otherwise>
                                                        </xsl:choose>
                                                    </xsl:variable>
                                                    <xsl:if test="($useVal!='') and ($useVal!='unrestricted') and
($useVal!='not filled') ">
                                                        <xsl:variable name="use_code">
                                                            <xsl:choose>
                                                                <xsl:when
test="contains($useCode,'copyright')">
                                                                    <xsl:value-
of select="'copyright'"/>
                                                                </xsl:when>
                                                                <xsl:when test="$useCode =
'restricted'">
                                                                    <xsl:value-
of select="'restricted'"/>
                                                                </xsl:when>
                                                                <xsl:otherwise>
                                                                    <xsl:value-
of select="'otherRestrictions'"/>
                                                                </xsl:otherwise>
                                                            </xsl:choose>
                                                        </xsl:if>
                                                    </xsl:choose>
                                                </xsl:for-each>
                                                </xsl:otherwise>
                                                </xsl:choose>
                                            </xsl:variable>
                                            <xsl:if test="($useVal!='') and ($useVal!='unrestricted') and
($useVal!='not filled') ">
                                                <xsl:variable name="use_code">
                                                    <xsl:choose>
                                                        <xsl:when
test="contains($useCode,'copyright')">
                                                            <xsl:value-
of select="'copyright'"/>
                                                        </xsl:when>
                                                        <xsl:when test="$useCode =
'restricted'">
                                                            <xsl:value-
of select="'restricted'"/>
                                                        </xsl:when>
                                                        <xsl:otherwise>
                                                            <xsl:value-
of select="'otherRestrictions'"/>
                                                        </xsl:otherwise>
                                                    </xsl:choose>
                                                </xsl:if>
                                            </xsl:choose>
                                        </xsl:variable>
                                        <xsl:variable name="useCode">
                                            <xsl:choose>
                                                <xsl:when test="contains($use,':')">
                                                    <xsl:value-of select="normalize-
space(substring-before($use,':'))"/>
                                                </xsl:when>
                                                <xsl:otherwise>
                                                    <xsl:value-of
select="$use"/>
                                                </xsl:otherwise>
                                            </xsl:choose>
                                        </xsl:variable>
                                        <xsl:if test="($useVal!='') and ($useVal!='unrestricted') and
($useVal!='not filled') ">
                                            <xsl:variable name="use_code">
                                                <xsl:choose>
                                                    <xsl:when
test="contains($useCode,'copyright')">
                                                        <xsl:value-
of select="'copyright'"/>
                                                    </xsl:when>
                                                    <xsl:when test="$useCode =
'restricted'">
                                                        <xsl:value-
of select="'restricted'"/>
                                                    </xsl:when>
                                                    <xsl:otherwise>
                                                        <xsl:value-
of select="'otherRestrictions'"/>
                                                    </xsl:otherwise>
                                                </xsl:choose>
                                            </xsl:if>
                                        </xsl:choose>
                                    </xsl:for-each>
                                </xsl:otherwise>
                                </xsl:choose>
                                </MD_RestrictionCode>
                                </accessConstraints>
                                </xsl:if>
                                </xsl:for-each>
```



```
        </xsl:variable>
    <useConstraints>
        <MD_RestrictionCode
codeList="{ $codelist }#MD_RestrictionCode"
        codeListValue="{ $use_code }">
            <xsl:choose>
                <xsl:when
test="contains($useVal, 'not available')">
                    <xsl:value-of select="$C3Grid_NO_ACCESS"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="$useVal"/>
                </xsl:otherwise>
            </xsl:choose>
        </MD_RestrictionCode>
    </useConstraints>
</xsl:if>
</xsl:for-each>

    <xsl:for-each select="/CERA2/DISTR/ROWSET/ROW">
        <xsl:if test="contains(ACCESS_CONSTRAINT_DESCR, 'not
available') or contains(USE_CONSTRAINT_DESCR, 'not available')">
            <otherConstraints>
                <gco:CharacterString>
                    <xsl:value-of select="$C3Grid_NO_ACCESS"/>
                </gco:CharacterString>
            </otherConstraints>
        </xsl:if>
    </xsl:for-each>

</MD_LegalConstraints>

<!-- spatialRepresentationTypeCode is 002 or grid for model data
others? -->
</resourceConstraints>
<xsl:for-each
select="/CERA2/TOPIC_ALL/ROWSET/ROW[contains(REFERENCE_METHOD, 'non
gridded')=false()]">
    <xsl:if test="contains(REFERENCE_METHOD, 'gridded')=true() ">
        <spatialRepresentationType>
            <MD_SpatialRepresentationTypeCode
codeList="{ $codelist }#MD_SpatialRepresentationTypeCode"
codeListValue="grid">grid</MD_SpatialRepresentationTypeCode>
        </spatialRepresentationType>
    </xsl:if>
</xsl:for-each>
<language>
    <LanguageCode codeList="{ $codelistML }#LanguageCode"
codeListValue="eng">eng</LanguageCode>
</language>
<topicCategory>
    <MD_TopicCategoryCode>
        <xsl:variable name="expSum"><xsl:value-of
select="/CERA2/ENTRY_EXP_SUM/ROWSET/ROW/SUMMARY"/></xsl:variable>
```



```

        <xsl:choose>
            <xsl:when test="contains($expSum,'OPA') or
contains($expSum,'NEMO')">
                <xsl:value-of select="'oceans'"/>
            </xsl:when>
            <xsl:when test="contains($expSum,'IPCC')">
                <xsl:value-of
select="'climatologyMeteorologyAtmosphere'"/>
            </xsl:when>
            <xsl:otherwise>
                <xsl:value-of
select="'climatologyMeteorologyAtmosphere'"/>
            </xsl:otherwise>
        </xsl:choose>
    </MD_TopicCategoryCode>
</topicCategory>
<environmentDescription>
    <gco:CharacterString>
        <xsl:choose>
            <xsl:when
test="( /CERA2/CHECK_EXT_EXP/ROWSET/ROW/CNT!=0) or
( /CERA2/CHECK_EXT_DS/ROWSET/ROW/CNT!=0) ">
                <xsl:value-of select="'DKRZ Archive Data'"/>
            </xsl:when>
            <xsl:when
test="( /CERA2/CHECK_CERA_EXP/ROWSET/ROW/CNT!=0) or
( /CERA2/CHECK_CERA_DS/ROWSET/ROW/CNT!=0) ">
                <xsl:value-of select="'CERA Database Data'"/>
            </xsl:when>
        </xsl:choose>
        <!--
        PROGRESS:<xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/PROGRESS_DESCR"/>
        <xsl:for-each select="/CERA2/CODE_ALL/ROWSET/ROW">
            <xsl:if test="position() = 1">#</xsl:if>
            TYPE:<xsl:value-of select="CODE_TYPE"/>
            ,NO:<xsl:value-of select="CODE_NUMBER"/>
            ,AC:<xsl:value-of select="CODE_ACRONYM"/>
            ,DESCR:<xsl:value-of select="CODE_DESCR"/>
            <xsl:if test="position() != last()"/></xsl:if>
        </xsl:for-each>
        -->
    </gco:CharacterString>
</environmentDescription>
    <xsl:for-each select="/CERA2/COVER/ROWSET/ROW">
        <xsl:variable name="id"><xsl:value-of
select="ENTRY_ID"/></xsl:variable>
        <extent>
            <!-- data specific information !!! -->
            <EX_Extent>
                <!-- description with CharacterString(?) -->
                <description>
                    <gco:CharacterString><xsl:if
test=".../.../SPATI/ROWSET/ROW/VER_SYS_DESCR">vertical_coordinate_sytem:</xs
l:if><xsl:value-of select=".../.../SPATI/ROWSET/ROW/VER_SYS_DESCR"/>

```



```

        <xsl:if
test="../../../SPATI/ROWSET/ROW/HOR_SYS_DESCR">horizontal_coordinate_sytem:</
xsl:if><xsl:value-of
select="../../../SPATI/ROWSET/ROW/HOR_SYS_DESCR"/></gco:CharacterString>
        </description>
        <!-- geographic (LON/LAT) coverage of data sets in experiment
-->
        <geographicElement>
          <EX_GeographicBoundingBox>
            <westBoundLongitude>
              <gco:Decimal>
                <xsl:value-of select="MIN_LON"/>
              </gco:Decimal>
            </westBoundLongitude>
            <eastBoundLongitude>
              <gco:Decimal>
                <xsl:value-of select="MAX_LON"/>
              </gco:Decimal>
            </eastBoundLongitude>
            <southBoundLatitude>
              <gco:Decimal>
                <xsl:value-of select="MIN_LAT"/>
              </gco:Decimal>
            </southBoundLatitude>
            <northBoundLatitude>
              <gco:Decimal>
                <xsl:value-of select="MAX_LAT"/>
              </gco:Decimal>
            </northBoundLatitude>
          </EX_GeographicBoundingBox>
        </geographicElement>

        <!--
                                P1Y2M3DT10H30M
                                0001-02-03T04:05:06Z

date:add(date:date-time(),'-PT0H')

<xsl:value-of select="START_DATE"/>
                                <xsl:call-template name="date:date">
                                <xsl:with-param name="date-time"
select="$tmBeginVVV" />
                                </xsl:call-template>

                                &#62; 0

                                <xsl:value-of select="'2000-11-10T00:00Z'"/>
                                <xsl:call-template name="date:date">
                                <xsl:with-param name="date-time"
select="$tmBeginVVV" />
                                </xsl:call-template>

                                <xsl:call-template name="date:add">

```



```

select="date:date()"/>
select="$tmBeginVVV"/>

<xsl:value-of select="date:date-time()"/>
-->
  <temporalElement>
    <EX_TemporalExtent>
      <extent>
        <gml:TimePeriod gml:id="extent">

          <xsl:if test="$ffname=''">

<xsl:variable name="tmBegin2">
  <xsl:value-of select="START_YEAR"/>
  <xsl:choose>
    <xsl:when test="string-
      <xsl:value-of select="'-0'"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="'-'"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:value-of select="START_MONTH"/>
  <xsl:choose>
    <xsl:when test="string-
      <xsl:value-of select="'-0'"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="'-'"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:value-of select="START_DAY"/>
  <xsl:value-of select="'T00:00:00'"/>

</xsl:variable>

  <xsl:variable name="tmBegin">
    <xsl:call-template name="date:add">
      <xsl:with-param name="date-time"

    <xsl:with-param name="duration"

    </xsl:call-template>
  </xsl:variable>

<xsl:variable name="tmEnd1">
  <xsl:value-of select="STOP_YEAR"/>
  <xsl:choose>
    <xsl:when test="string-
length(string(START_MONTH))=1">
length(string(START_DAY))=1">
length(string(STOP_MONTH))=1">

```




```

        <xsl:value-of select="$tmEnd2"/>
    </xsl:otherwise>
        </xsl:choose>
    </xsl:variable>

<xsl:variable name="valBegin"><xsl:value-of select="translate($tmBegin, '
','')"/></xsl:variable>

<xsl:variable name="valEnd"><xsl:value-of select="translate($tmEnd, '
','')"/></xsl:variable>
        <gml:beginPosition><xsl:value-of
select="$valBegin"/></gml:beginPosition>
        <gml:endPosition><xsl:value-of
select="$valEnd"/></gml:endPosition>

<xsl:variable name="tmStep">
        <xsl:choose>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_YEAR!=0"><xsl:value-of
select="/CERA2/TIMESTEP/ROWSET/ROW/STEP_YEAR"/></xsl:when>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_MONTH!=0"><xsl:value-of
select="/CERA2/TIMESTEP/ROWSET/ROW/STEP_MONTH"/></xsl:when>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_DAY!=0"><xsl:value-of
select="/CERA2/TIMESTEP/ROWSET/ROW/STEP_DAY"/></xsl:when>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_HOUR!=0"><xsl:value-of
select="/CERA2/TIMESTEP/ROWSET/ROW/STEP_HOUR"/></xsl:when>
            <xsl:otherwise>0</xsl:otherwise>
        </xsl:choose>
    </xsl:variable>

<xsl:variable name="tmUnit">
        <xsl:choose>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_YEAR!=0">year</xsl:when>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_MONTH!=0">month</xsl:when>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_DAY!=0">day</xsl:when>
            <xsl:when
test="/CERA2/TIMESTEP/ROWSET/ROW/STEP_HOUR!=0">hour</xsl:when>
            <xsl:otherwise><xsl:value-of
select="$MyDummy"/></xsl:otherwise>
        </xsl:choose>
    </xsl:variable>

```



```

<xsl:variable name="valStep"><xsl:value-of select="translate($tmStep, '
', '')"/></xsl:variable>
                                <!--
                                <xxx><xsl:value-of select="$valStep"/></xxx>
                                -->

<xsl:variable name="valUnit">
    <xsl:value-of select="$tmUnit"/>
</xsl:variable>

<xsl:variable name="valStepCnt">
    <xsl:call-template name="calcStepCnt">
        <xsl:with-param name="start" select="$tmBegin"/>
        <xsl:with-param name="end" select="$tmEnd"/>
        <xsl:with-param name="tmStepDur"
select="$tmStep"/>
    </xsl:call-template>
</xsl:variable>

    <gml:timeInterval
        unit="{ $valUnit }"
        factor="-1"
        radix="{ $tmStep }"><xsl:value-of
select="$valStepCnt"/></gml:timeInterval>
    </xsl:if>
    <xsl:if test="$ffname!=''">
        <xsl:for-each
select="/CERA2/ENTRY_ALL_EXT/ROWSET/ROW">
            <xsl:variable name="tmBegin2">
                <xsl:choose>
                    <xsl:when test="string-
length(string(START_YEAR))=1">
                        <xsl:value-of
select="'000'"/>
                    </xsl:when>
                    <xsl:when test="string-
length(string(START_YEAR))=2">
                        <xsl:value-of
select="'00'"/>
                    </xsl:when>
                    <xsl:when test="string-
length(string(START_YEAR))=3">
                        <xsl:value-of
select="'0'"/>
                </xsl:choose>
            </xsl:variable>

```




```

select="$tmBegin" />
name="duration" select="translate(string($tmDuration1),' ','')" />
name="date-time" select="$tmEnd1" />
name="duration" select="'-PT1H'" />
test="contains($tmEnd1,'T00:00')">
template name="date:add">
name="date-time" select="$tmEnd1" />
name="duration" select="'-PT1H'" />
<xsl:with-param name="date-time"
<xsl:with-param
</xsl:call-template>
</xsl:variable>
<xsl:variable name="tmEnd">
<xsl:choose>
<xsl:when
<xsl:call-
<xsl:with-param
<xsl:with-param
</xsl:call-template>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="$tmEnd1"/>
</xsl:otherwise>
</xsl:choose>
</xsl:variable>
<xsl:variable name="valBegin"><xsl:value-of
select="translate($tmBegin,' ','')"/></xsl:variable>
<xsl:variable name="valEnd"><xsl:value-of
select="translate($tmEnd,' ','')"/></xsl:variable>
<xsl:if test="contains($valBegin,'Z')">
<gml:beginPosition><xsl:value-of
select="substring-before($valBegin,'Z')"/></gml:beginPosition>
</xsl:if>
<xsl:if test="not(contains($valBegin,'Z'))">
<gml:beginPosition><xsl:value-of
select="$valBegin"/></gml:beginPosition>
</xsl:if>
<xsl:if test="contains($valEnd,'Z')">
<gml:endPosition><xsl:value-of
select="substring-before($valEnd,'Z')"/></gml:endPosition>
</xsl:if>
<xsl:if test="not(contains($valEnd,'Z'))">
<gml:endPosition><xsl:value-of
select="$valEnd"/></gml:endPosition>

```



```

</xsl:if>

    <xsl:variable name="tmStep1">
        <xsl:choose>
            <xsl:when
test="STEP_YEAR!=0"><xsl:value-of select="STEP_YEAR"/></xsl:when>
            <xsl:when
test="STEP_MONTH!=0"><xsl:value-of select="STEP_MONTH"/></xsl:when>
            <xsl:when
test="STEP_DAY!=0"><xsl:value-of select="STEP_DAY"/></xsl:when>
            <xsl:when
test="STEP_HOUR!=0"><xsl:value-of select="STEP_HOUR"/></xsl:when>
        <xsl:otherwise>0</xsl:otherwise>
        </xsl:choose>

    </xsl:variable>

    <xsl:variable name="tmUnit1">
        <xsl:choose>
            <xsl:when
test="STEP_YEAR!=0">year</xsl:when>
            <xsl:when
test="STEP_MONTH!=0">month</xsl:when>
            <xsl:when
test="STEP_DAY!=0">day</xsl:when>
            <xsl:when
test="STEP_HOUR!=0">hour</xsl:when>
        <xsl:otherwise><xsl:value-of select="$MyDummy"/></xsl:otherwise>
        </xsl:choose>

    </xsl:variable>

    <xsl:variable name="tmStepCnt">
        <xsl:value-of select="TARGET_STEP_COUNT"/>
    </xsl:variable>

    <xsl:variable name="stepStepCnt">
        <xsl:value-of
select="TARGET_STEP_COUNT"/>
    </xsl:variable>

    <xsl:variable name="durTot">
        <xsl:call-template
name="date:difference">
            <xsl:with-param name="start"
select="$tmBegin"/>

```



```
select="$tmEnd"/>
<xsl:with-param name="end"
</xsl:call-template>
</xsl:variable>

<xsl:variable name="durTotVal">
<xsl:choose>
<xsl:when test="floor($tmStep1 div $stepStepCnt)=1">
<xsl:value-of select="'1#'" /><xsl:value-of
select="$tmUnit1"/>
</xsl:when>
<xsl:when test="floor($tmStep1 div $stepStepCnt)=0">
<xsl:call-template
name="calcStepData">
<xsl:with-param
name="start" select="$tmEnd"/>
<xsl:with-param
name="end" select="$tmBegin"/>
<xsl:with-param
name="tmStepCnt" select="$tmStepCnt"/>
<xsl:with-param
name="stepStepCnt" select="$stepStepCnt"/>
<xsl:with-param
name="tmStepP" select="$tmStep1"/>
<xsl:with-param
name="tmUnit" select="$tmUnit1"/>
</xsl:call-template>
</xsl:when>
</xsl:choose>
</xsl:variable>

<xsl:variable name="tmUnit">
<xsl:value-of select="substring-after($durTotVal,'#')"/>
</xsl:variable>

<xsl:variable name="tmStep">
<xsl:value-of select="substring-before($durTotVal,'#')"/>
</xsl:variable>

<!--
-->

<xsl:if test="$debug='show'">
```



```

        <xsl:value-of select="$debugInfo"/><xsl:value-of select="$durTotVal"/>
    </xsl:if>

        <gml:timeInterval unit="{ $tmUnit}" factor="-1"
radix="{ $tmStep}"><xsl:value-of select="$stepStepCnt"/></gml:timeInterval>

                </xsl:for-each>

                </xsl:if>

                </gml:TimePeriod>
        </extent>
    </EX_TemporalExtent>
</temporalElement>

        <xsl:for-each select="/CERA2/COVER_hPa/ROWSET/ROW">
            <xsl:if test="MINIMUM_ALT and MAXIMUM_ALT">
<verticalElement>
            <EX_VerticalExtent id="{ $verticalElementPrefix}hPa">
                <minimumValue>
                    <gco:Real>
                        <xsl:value-of select="MINIMUM_ALT"/>
                    </gco:Real>
                </minimumValue>
                <maximumValue>
                    <gco:Real>
                        <xsl:value-of select="MAXIMUM_ALT"/>
                    </gco:Real>
                </maximumValue>

                <xsl:variable
name="vertC3CRS">
                    <xsl:value-of
select="'vertCRS.standardPressureLevel'"/>
                </xsl:variable>
                <xsl:variable
name="topicList">
                    <xsl:value-of select="'add_info,'"/>
                    <!--
                <xsl:for-each
select="/CERA2/TOPIC_ALL_m_beneath/ROWSET/ROW">
                    <xsl:value-of
select="TOPIC_NAME"/><xsl:value-of select="','"/>
                </xsl:for-each>
                    -->
                </xsl:variable>

                <verticalCRS
xlink:href="{ $verticalCRSTemplateURL}#xpointer(//*[@gml:id='{ $vertC3CRS}'])"
                    xlink:title="CRS for Height"
                    xlink:role="{substring-
before(concat($topicList,'#'),'','#')}">
                </EX_VerticalExtent>
            </verticalElement>

```



```

        </xsl:if>
      </xsl:for-each>
      <xsl:for-each select="/CERA2/COVER_m/ROWSET/ROW">
        <xsl:if test="MINIMUM_ALT and MAXIMUM_ALT">
      <verticalElement>
        <EX_VerticalExtent id="{verticalElementPrefix}m">
          <minimumValue>
            <gco:Real>
              <xsl:value-of select="MINIMUM_ALT"/>
            </gco:Real>
          </minimumValue>
          <maximumValue>
            <gco:Real>
              <xsl:value-of select="MAXIMUM_ALT"/>
            </gco:Real>
          </maximumValue>
          <xsl:variable
name="vertC3CRS">
            <xsl:value-of
select="'vertCRS.depthBeneathSurface'"/>
          </xsl:variable>
          <xsl:variable
name="topicList">
            <xsl:value-of select="'add_info,'"/>
            <!--
          <xsl:for-each
select="/CERA2/TOPIC_ALL_m_beneath/ROWSET/ROW">
            <xsl:value-of
select="TOPIC_NAME"/><xsl:value-of select="','"/>
          </xsl:for-each>
          -->
          </xsl:variable>
          <verticalCRS
xlink:href="{verticalCRSTemplateURL}#xpointer(//*[@gml:id='{verticalC3CRS}'])"
            xlink:title="CRS for Height"
            xlink:role="{substring-
before(concat($topicList,'#'),',#')}">
          </EX_VerticalExtent>
        </verticalElement>
      </xsl:if>
    </xsl:for-each>

    </EX_Extent>
  </extent>
</xsl:for-each>
<!--
<supplementalInformation>
  <gco:CharacterString><xsl:value-of
select="$MyDummy"/></gco:CharacterString>
</supplementalInformation>
-->

```



```
</MD_DataIdentification>
<!--
</identificationInfo>
</xsl:for-each>
-->
</xsl:template>

<xsl:template name="printCitedResponsibleParty">
<xsl:for-each select="/CERA2/CITATION/ROWSET/ROW">
  <citedResponsibleParty>
    <CI_ResponsibleParty>
      <individualName>
        <gco:CharacterString>
          <xsl:value-of select="NAME"/>
        </gco:CharacterString>
      </individualName>
      <organisationName>
        <gco:CharacterString>
          <xsl:value-of
select="INSTITUTE_NAME"/>
        </gco:CharacterString>
      </organisationName>
      <contactInfo>
        <CI_Contact>
          <onlineResource>
            <CI_OnlineResource>
              <linkage>
                <URL>
                  <xsl:value-
of select="INST_URL"/>
                </URL>
              </linkage>
            </CI_OnlineResource>
            <function>
              <CI_OnLineFunctionCode
codeList="http://www.isotc211.org/2005/resources/Codelist/gmxCodelists.xml#CI
_OnLineFunctionCode"
codeListValue="http://www.isotc211.org/2005/resources/Codelist/gmxCodelists.x
ml#CI_OnLineFunctionCode_information">information
            </CI_OnLineFunctionCode>
          </function>
        </CI_OnlineResource>
      </onlineResource>
    </CI_Contact>
  </contactInfo>
  <xsl:call-template name="printRole">
    <xsl:with-param name="role" select="CONTACT_TYPE"/>
  </xsl:call-template>
</CI_ResponsibleParty>
</citedResponsibleParty>
</xsl:for-each>
</xsl:template>
```



```
<xsl:template name="calcStepCnt">
  <!-- -P30DT23H -->
  <xsl:param name="start"/>
  <xsl:param name="end"/>
  <!-- tmStepLen e.g. 6,12,... -->
  <xsl:param name="tmStepDur"/>
  <xsl:variable name="durTot">
    <xsl:call-template name="date:difference">
      <xsl:with-param name="start" select="$start"/>
      <xsl:with-param name="end" select="$end"/>
    </xsl:call-template>
  </xsl:variable>
  WWW<xsl:value-of select="$durTot"/>WWW<xsl:value-of
select="( (number(substring-before(substring-after($durTot,'P'),'D')) * 24)
div $tmStepDur)"/>WWW
</xsl:template>
```

```
<xsl:template name="calcStepData">
  <!-- -P30DT23H -->
  <xsl:param name="start"/>
  <xsl:param name="end"/>
  <!-- sum stepcount e.g. 124 -->
  <xsl:param name="tmStepCnt"/>
  <!-- taget_step_count -->
  <xsl:param name="stepStepCnt"/>
  <!-- 1 -->
  <xsl:param name="tmStepP"/>
  <!-- month -->
  <xsl:param name="tmUnit"/>

  <xsl:variable name="durTot">
    <xsl:call-template name="date:difference">
      <xsl:with-param name="start" select="$start"/>
      <xsl:with-param name="end" select="$end"/>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="durTotY">
    <xsl:call-template name="date:difference">
      <xsl:with-param name="start" select="substring-
before($start,'-')"/>
      <xsl:with-param name="end" select="substring-before($end,'-')"/>
    </xsl:call-template>
  </xsl:variable>

  <xsl:variable name="durTotYval">
    <xsl:value-of select="ceiling(substring-before(substring-
after($durTotY,'P'),'Y'))"/>
  </xsl:variable>

  <!--
  <xsl:variable name="durTotD">
    <xsl:call-template name="date:difference">
```



```
<xsl:with-param name="start" select="substring-before(substring-
before($start,'T'),'-')"/>
  <xsl:with-param name="end" select="substring-before(substring-
before($end,'T'),'-')"/>
</xsl:call-template>
</xsl:variable>

<xsl:variable name="durTotDval">
  <xsl:value-of select="ceiling(substring-before(substring-
after($durTotD,'P'),'D'))"/>
</xsl:variable>
<xsl:value-of select="(30 * 24) div $stepStepCnt"/>
  <xsl:value-of select="$durTotDval"/>

-->
<xsl:if test="$debug='show'">
  <Param>
    OAA<xsl:value-of select="$durTot"/>
    OBB<xsl:value-of select="($tmStepCnt div $durTotYval)
div $stepStepCnt"/>
    OCC<xsl:value-of select="$tmStepCnt"/>
    ODD<xsl:value-of select="$stepStepCnt"/>
    OEE<xsl:value-of select="$tmStepP"/>
    OFF<xsl:value-of select="$tmUnit"/>OZZ
  </Param>
</xsl:if>

  <xsl:variable name="durY">
    <xsl:if test="contains($durTot,'Y')">
      <xsl:value-of select="substring-before(substring-
after($durTot,'P'),'Y')"/>
    </xsl:if>
  </xsl:variable>

  <xsl:variable name="durM">
    <xsl:if test="contains($durTot,'M')">
      <xsl:if test="contains($durTot,'Y')">
        <xsl:value-of select="substring-before(substring-
after($durTot,'Y'),'M')"/>
      </xsl:if>
      <xsl:if test="not(contains($durTot,'Y'))">
        <xsl:if test="contains($durTot,'M')">
          <xsl:value-of select="substring-before(substring-
after($durTot,'P'),'M')"/>
        </xsl:if>
      </xsl:if>
    </xsl:if>
  </xsl:variable>

  <xsl:variable name="durD">
    <xsl:if test="contains($durTot,'D')">
      <xsl:if test="contains($durTot,'P') and
not(contains($durTot,'Y')) and not(contains($durTot,'M'))">
        <xsl:value-of select="substring-before(substring-
after($durTot,'P'),'D')"/>
      </xsl:if>
    </xsl:if>
  </xsl:variable>
```



```

        </xsl:if>
        <xsl:if test="contains($durTot, 'Y')">
            <xsl:value-of select="substring-before(substring-
after($durTot, 'Y'), 'D')"/>
        </xsl:if>
        <xsl:if test="contains($durTot, 'M')">
            <xsl:if test="contains($durTot, 'D')">
                <xsl:value-of select="substring-before(substring-
after($durTot, 'M'), 'D')"/>
            </xsl:if>
        </xsl:if>
    </xsl:if>
</xsl:variable>

    <!--
    <xsl:variable name="durH">
        <xsl:choose>
            <xsl:when test="contains($durTot, 'T')">
                <xsl:value-of select="substring-before(substring-
after($durTot, 'T'), 'H')"/>
            </xsl:when>
            <xsl:otherwise>0</xsl:otherwise>
        </xsl:choose>
    </xsl:variable>

    <xsl:variable name="durHcorr">
        <xsl:value-of select="24 - ($durH mod 6)"/>
    </xsl:variable>

        <xsl:variable name="durHval">
            <xsl:choose>
                <xsl:when test="string(number($durD))!='NaN' and
string(number($durM))!='NaN' and string(number($durY))!='NaN'">
                    <xsl:value-of select="ceiling((((($durD + $durM + $durY) *
24) + $durH + $durHcorr) div $tmStepCnt)"/>
                </xsl:when>
                <xsl:when test="string(number($durD))!='NaN' and
string(number($durM))!='NaN'">
                    <xsl:value-of select="ceiling((((($durD + $durM) * 24) + $durH +
$durHcorr) div $tmStepCnt)"/>
                </xsl:when>
                <xsl:when test="string(number($durD))!='NaN'">
                    <xsl:value-of select="ceiling(((($durD * 24) + $durH + $durHcorr)"/>
                </xsl:when>
            </xsl:choose>
        </xsl:variable>

        <xsl:value-of select="($durD * 24) + $durHcorr"/> = 715
        AA
        <xsl:value-of select="$durH"/>
    X
        <xsl:value-of select="$durHcorr"/>
        Q
        <xsl:value-of select="ceiling($durHval div $tmStepCnt)"/>
        Q
        <xsl:value-of select="$durHval"/>

```



```
-->
<xsl:if test="$debug='show'">
  A
  <xsl:value-of select="$durD"/>
  B
  <xsl:value-of select="($durD * 24)"/>
  C
  <xsl:value-of select="($durD * 24) + 24"/>
  D
  <xsl:value-of select="$tmStepCnt"/>
  Y
  <xsl:value-of select="$durD"/>
  Q
  <xsl:value-of select="floor($tmStepP div $stepStepCnt)"/>
  QQ
  <xsl:value-of select="round((( $durD * 24) + 24) div
$tmStepCnt)"/>
  QQ
  <xsl:value-of select="ceiling((( $durD * 24) + 24) div
$tmStepCnt)"/>
  QQ
  <xsl:value-of select="floor((( $durD * 24) + 24) div
$tmStepCnt)"/>
  ORG<xsl:value-of select="$tmUnit"/>NEW
</xsl:if>
<xsl:variable name="tmStepValTst">
  <xsl:value-of select="round((( $durD * 24) + 24) div
$tmStepCnt)"/>
</xsl:variable>
  <xsl:variable name="tmStepVal">
    <xsl:choose>
      <xsl:when test="(($tmStepValTst mod 6)=0) or
(($tmStepValTst mod 12)=0) or (($tmStepValTst mod 24)=0)">
        <xsl:value-of select="$tmStepValTst"/>
      </xsl:when>
      <xsl:otherwise><xsl:value-of select="ceiling((( $durD
* 24) + 24) div $tmStepCnt)"/></xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable name="tmStep">
    <xsl:choose>
      <xsl:when test="(($tmStepVal mod 6)=0) or
(($tmStepVal mod 12)=0) or (($tmStepVal mod 24)=0)">
        <xsl:value-of select="$tmStepVal"/>
      </xsl:when>
      <xsl:otherwise><xsl:value-of select="($tmStepCnt div
$durTotYval) div $stepStepCnt"/></xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:value-of select="$tmStep"/>
  <xsl:value-of select="'#'"/>
  <xsl:choose>
    <xsl:when test="$tmUnit='year'">
      <xsl:if test="(($tmStep mod 6)=0) or (($tmStep mod
12)=0) or (($tmStep mod 24)=0)">
```



```

        <xsl:value-of select="'hour'"/>
    </xsl:if>
</xsl:when>
<xsl:when test="$tmUnit='month'">
    <xsl:if test="(($tmStep mod 6)=0) or (($tmStep mod
12)=0) or (($tmStep mod 24)=0)">
        <xsl:value-of select="'hour'"/>
    </xsl:if>
</xsl:when>
<xsl:when test="$tmUnit='hour'">
    <xsl:if test="((tmStep mod 6)=0) or (($tmStep mod
12)=0) or (($tmStep mod 24)=0)">
        <xsl:value-of select="$tmUnit"/>
    </xsl:if>
</xsl:when>
<xsl:otherwise>not known</xsl:otherwise>
</xsl:choose>

</xsl:template>

<xsl:template name="printProviderInfo">
    <xsl:param name="function"/>
    <individualName>
        <gco:CharacterString>
            <xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/LAST_NAME"/>
            <xsl:if
test="(/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/LAST_NAME!='
') and
(/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/FIRST_NAME!='')">
                <xsl:text>, </xsl:text>
            </xsl:if>
            <xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/FIRST_NAME"
/>
        </gco:CharacterString>
    </individualName>
    <organisationName>
        <gco:CharacterString>
            <xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/INSTITUTE_N
AME"/>
            <xsl:if
test="(/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/INSTITUTE_NA
ME!='') and
(/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/DEPARTMENT_NAME!='
')">
                <xsl:text>, </xsl:text>
            </xsl:if>
            <xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE,$function)]/DEPARTMENT_
NAME"/>
        </gco:CharacterString>
    </organisationName>

```



```
<contactInfo>
  <CI_Contact>
<!--
-->
    <address>
      <CI_Address>
        <electronicMailAddress>
          <gco:CharacterString>
            <xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE, $function)]/EMAIL"/>
          </gco:CharacterString>
        </electronicMailAddress>
      </CI_Address>
    </address>
    <onlineResource>
      <CI_OnlineResource>
        <linkage>
          <URL>
            <xsl:value-of
select="/CERA2/CONTA/ROWSET/ROW[contains (CONTACT_TYPE, $function)]/INST_URL"/>
          </URL>
        </linkage>
      </function>
      <CI_OnLineFunctionCode
codeList="{ $codelist}#CI_OnLineFunctionCode"
codeListValue="information">information</CI_OnLineFunctionCode>
    </function>
  </CI_OnlineResource>
</onlineResource>
</CI_Contact>
</contactInfo>
</xsl:template>

<xsl:template name="printRole">
  <xsl:param name="role"/>
  <xsl:if test="$debug='showTplNames'">
    <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="printRole"/>
  </xsl:if>
  <role>
    <xsl:choose>
      <xsl:when test="($role = 'distributor') or ($role =
'originator') or ($role = 'owner') or ($role = 'publisher') or ($role =
'author')">
        <CI_RoleCode codeList="{ $codelist}#CI_RoleCode"
codeListValue="{ $role}"><xsl:value-of select="$role"/></CI_RoleCode>
      </xsl:when>
      <xsl:when test="$role = 'investigator'">
        <CI_RoleCode codeList="{ $codelist}#CI_RoleCode"
codeListValue="principalInvestigator"><xsl:value-of
select="$role"/></CI_RoleCode>
      </xsl:when>
      <xsl:when test="$role = 'DOI_author'">
```



```
        <CI_RoleCode codeList="{ $codelist }#CI_RoleCode"
codeListValue="author"><xsl:value-of select="$role"/></CI_RoleCode>
        </xsl:when>
        <xsl:otherwise>
        <CI_RoleCode codeList="{ $codelist }#CI_RoleCode"
codeListValue="pointOfContact"><xsl:value-of select="$role"/></CI_RoleCode>
        </xsl:otherwise>
    </xsl:choose>
    </role>
</xsl:template>
```

<!-- distributionFormat are these formats which could be provided directly by data source for submission -->

```
<xsl:template name="distributionInfo">
    <MD_Distribution>
        <xsl:call-template name="printFormatDist"/>
        <xsl:call-template name="distributorInfo"/>
    <!--
    <transferOptions>
        <MD_DigitalTransferOptions>
            <transferSize>
                <gco:Real>
                    <xsl:value-of select="/CERA2/EXP_SIZE/ROWSET/ROW/TOTAL_SIZE"/>
                </gco:Real>
            </transferSize>
            <xsl:call-template name="distributionOnline"/>
        </MD_DigitalTransferOptions>
    </transferOptions>
    -->
</MD_Distribution>
</xsl:template>
```

```
<xsl:template name="distributorInfo">
    <distributor>
        <MD_Distributor>
            <distributorContact>
                <CI_ResponsibleParty>
                    <xsl:call-template name="printProviderInfo">
                        <xsl:with-param name="function" select="'metadata'"/>
                    </xsl:call-template>
                    <xsl:call-template name="printRole">
                        <xsl:with-param name="role" select="'distributor'"/>
                    </xsl:call-template>
                </CI_ResponsibleParty>
            </distributorContact>
            <distributorTransferOptions>
                <MD_DigitalTransferOptions>
                    <transferSize>
                        <gco:Real>
                            <xsl:value-of
select="/CERA2/EXP_SIZE/ROWSET/ROW/TOTAL_SIZE"/>
                        </gco:Real>
                    </transferSize>
                    <xsl:call-template name="distributionOnline"/>
                </MD_DigitalTransferOptions>
            </distributorTransferOptions>
        </MD_Distributor>
    </distributor>
</xsl:template>
```



```
        </MD_DigitalTransferOptions>
    </distributorTransferOptions>
</MD_Distributor>
</distributor>
</xsl:template>

<xsl:template name="distributorInfoData">
    <distributor>
        <MD_Distributor>
            <distributorContact>
                <CI_ResponsibleParty>
                    <xsl:call-template name="printProviderInfo">
                        <xsl:with-param name="function" select="'metadata'"/>
                    </xsl:call-template>
                    <xsl:call-template name="printRole">
                        <xsl:with-param name="role" select="'distributor'"/>
                    </xsl:call-template>
                </CI_ResponsibleParty>
            </distributorContact>
            <xsl:if test="$ffname!=''">
                <xsl:variable name="path1"
select="/CERA2/STORAGE/ROWSET/ROW/STORAGE1"/>
                <xsl:variable name="path2"
select="/CERA2/STORAGE/ROWSET/ROW/STORAGE2"/>
                <xsl:variable name="searchchr" select="'&#47;'"/>
                <xsl:variable name="archivePrefix">
                    <xsl:choose>
                        <xsl:when
test="contains($path1,'archiveXXX')">dkrz</xsl:when>
                        <xsl:when
test="contains($path1,'schauerXXX')">schauer</xsl:when>
                        <xsl:when
test="contains($path1,'crossXXX')">cross</xsl:when>
                        <xsl:otherwise><xsl:value-of
select="$path1"/></xsl:otherwise>
                    </xsl:choose>
                </xsl:variable>
                <xsl:for-each select="/CERA2/ENTRY_ALL_EXT/ROWSET/ROW">
                    <xsl:variable name="path">
                        <xsl:choose>
                            <xsl:when
test="substring(TARGET,1,1)='/'">
                                <xsl:value-of
select="substring(TARGET,2)"/>
                            </xsl:when>
                            <xsl:otherwise>
                                <xsl:value-of select="TARGET"/>
                            </xsl:otherwise>
                        </xsl:choose>
                    </xsl:variable>
                    <distributorTransferOptions>
                        <MD_DigitalTransferOptions>
                            <transferSize>
                                <gco:Real>
                                    <xsl:value-of select="DATASET_SIZE"/>
                                </gco:Real>
                            </transferSize>
                        </MD_DigitalTransferOptions>
                    </distributorTransferOptions>
                </xsl:for-each>
            </xsl:if>
        </MD_Distributor>
    </distributor>
</xsl:template>
```



```
</gco:Real>
</transferSize>
  <onLine>
    <CI_OnlineResource>
      <linkage>
        <URL>

      <xsl:value-of select="$distributorPrefix"/><xsl:value-of
select="$archivePrefix"/>:<xsl:value-of
select="concat(concat($path2, '/'), $path)"/>
        </URL>
      </linkage>
      <protocol>

    <gco:CharacterString>local</gco:CharacterString>
      </protocol>
      <name>
        <gco:CharacterString>

      <xsl:value-of select="$C3GridDataAccessInfo"/>
        </gco:CharacterString>
      </name>
      <function>

        <CI_OnLineFunctionCode
codeList="{ $odelist }#CI_OnLineFunctionCode"
codeListValue="information">information</CI_OnLineFunctionCode>
      </function>
      </CI_OnlineResource>
    </onLine>

    <xsl:call-template
name="webmdsEndpointOnline"/>
      </MD_DigitalTransferOptions>
    </distributorTransferOptions>
  </xsl:for-each>
</xsl:if>
  <xsl:if test="$ffname=''">
    <distributorTransferOptions>
      <MD_DigitalTransferOptions>
        <transferSize>
          <gco:Real>

            <xsl:value-of
select="/CERA2/FORMAT_DS/ROWSET/ROW/DATASET_SIZE"/>
          </gco:Real>
        </transferSize>
        <onLine>
          <CI_OnlineResource>
            <linkage>
              <URL>

            <xsl:value-
of select="$distributorPrefix"/><xsl:value-of
select="$ceraPrefix"/>:<xsl:value-of select="$entryAcro"/>
              </URL>
            </linkage>
            <protocol>
          <gco:CharacterString>local</gco:CharacterString>
```



```
</protocol>
  <name>
    <gco:CharacterString>
of select="$C3GridDataAccessInfo"/>
    </gco:CharacterString>
  </name>
  <!--
    <applicationProfile>
      <gco:CharacterString><xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ACCURACY_REPORT"/></gco:CharacterString>
    </applicationProfile>

    <name>
      <gco:CharacterString>
of select="$distributorPrefix"/><xsl:value-of
select="$ceraPrefix"/>:<xsl:value-of select="$entryAcro"/>
      </gco:CharacterString>
    </name>
  -->
  <function>
    <CI_OnLineFunctionCode
codeList="{ $codelist }#CI_OnLineFunctionCode"
codeListValue="information">information</CI_OnLineFunctionCode>
  </function>
  </CI_OnlineResource>
</onLine>
  </MD_DigitalTransferOptions>
</distributorTransferOptions>
</xsl:if>
  </MD_Distributor>
</distributor>
</xsl:template>

  <!-- distributionFormat are these formats which could be provided directly
by data source for submission -->
  <xsl:template name="distributionInfoData">
    <xsl:if test="$debug='showTplNames'">
      <xsl:value-of select="$debugInfoTpl"/><xsl:value-of
select="distributionInfoData"/>
    </xsl:if>
    <xsl:call-template name="printFormatDistData"/>
    <xsl:call-template name="distributorInfoData"/>
  </xsl:template>

  <xsl:template name="webmdsEndpointOnline">
    <onLine>
      <CI_OnlineResource>
        <linkage>
          <URL>
            <xsl:value-of select="$webmdsEndpoint"/>
          </URL>
        </linkage>
      </protocol>
```



```

        <gco:CharacterString>
            <xsl:value-of select="$webmdsProtocol"/>
        </gco:CharacterString>
    </protocol>
    <name>
        <gco:CharacterString><xsl:value-of
select="$webmdsName"/></gco:CharacterString>
    </name>
    <description>
        <gco:CharacterString><xsl:value-of
select="$webmdsDescr"/> </gco:CharacterString>
    </description>
    <function>
        <CI_OnLineFunctionCode
codeList="http://wis.wmo.int/2006/catalogues/gmxCodelists.xml#CI_OnLineFunci
onCode"

        codeListValue="order">order</CI_OnLineFunctionCode>
        </function>
    </CI_OnlineResource>
</onLine>
</xsl:template>

<xsl:template name="Online-IFM-GEOMAR-DODS">
    <onLine>
        <CI_OnlineResource>
            <linkage>
                <URL>http://dods.ifm-
geomar.de:8080/thredds/catalog/05/KAB042/output/catalog.html</URL>
            </linkage>
            <protocol>
                <gco:CharacterString>http</gco:CharacterString>
            </protocol>
            <name>
                <gco:CharacterString>C3Grid_Opendap</gco:CharacterString>
            </name>
            <description>
                <gco:CharacterString>OPeNDAP Dataset Access Server IFM-
GEOMAR</gco:CharacterString>
            </description>
            <function>
                <CI_OnLineFunctionCode
codeListValue="http://wis.wmo.int/2006/catalogues/gmxCodelists.xml#CI_OnLineF
unctionCode_download"

codeList="http://wis.wmo.int/2006/catalogues/gmxCodelists.xml#CI_OnLineFunci
onCode">download</CI_OnLineFunctionCode>
        </function>
    </CI_OnlineResource>
</onLine>
```



```
</xsl:template>

<xsl:template name="Online-DKRZ">
  <onLine>
    <CI_OnlineResource>
      <linkage>
        <URL>
          <xsl:value-of select="$webserviceEndpoint"/>
        </URL>
      </linkage>
      <protocol>
        <gco:CharacterString>
          <xsl:value-of
select="$webserviceProtocol"/>
        </gco:CharacterString>
      </protocol>
      <!--
        <xsl:variable name="model">
          <xsl:value-of
select="/CERA2/KEY_C/ROWSET/ROW[contains($modelList,GENERAL_KEY)]/GENERAL_KEY
"/>
          </xsl:variable>
          <xsl:variable name="data_res">
            <xsl:choose>
              <xsl:when
test="contains (/CERA2/ENTRY/ROWSET/ROW/ACCURACY_REPORT,'data_resolution')">
                <xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ACCURACY_REPORT"/>
              </xsl:when>
              <xsl:when
test="contains (/CERA2/ENTRY/ROWSET/ROW/HORIZONTAL_ACC_REPORT,[''])">
                <xsl:value-of select="substring-
before(substring-
after (/CERA2/ENTRY/ROWSET/ROW/HORIZONTAL_ACC_REPORT,':'),']')"/>
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="'no glue'"/>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:variable>
          <xsl:variable name="applicationPro">
            <xsl:choose>
              <xsl:when
test="contains (/CERA2/ENTRY/ROWSET/ROW/ACCURACY_REPORT,'data_resolution')">
                <xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ACCURACY_REPORT"/>
              </xsl:when>
              <xsl:otherwise>
                (model,<xsl:value-of
select="$model"/>) (data_resolution,<xsl:value-of
select="translate($data_res,'/',' ')/>)
              </xsl:otherwise>
            </xsl:choose>
          </xsl:variable>
        </gco:CharacterString>
      </protocol>
    </CI_OnlineResource>
  </onLine>
</xsl:template>
```



```

                                (ExpId,<xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_ID"/>)
                                </xsl:variable>
    <applicationProfile>
      <gco:CharacterString><xsl:value-of
select="$applicationPro"/></gco:CharacterString>
    </applicationProfile>
      -->
      <name>
        <gco:CharacterString><xsl:value-of
select="$webserviceName"/></gco:CharacterString>
      </name>
      <description>
        <gco:CharacterString><xsl:value-of
select="$webserviceDescr"/></gco:CharacterString>
      </description>
      <function>
        <CI_OnLineFunctionCode
codeList="{ $codelist}#CI_OnLineFunctionCode"
codeListValue="order">order</CI_OnLineFunctionCode>
        </function>
      </CI_OnlineResource>
    </onLine>
    <onLine>
      <CI_OnlineResource>
        <linkage>
          <URL><xsl:value-of select="$workspaceDir"/></URL>
        </linkage>
        <protocol>
          <gco:CharacterString>
            <xsl:value-of select="$workspaceProtocol"/>
          </gco:CharacterString>
        </protocol>
        <name>
          <gco:CharacterString><xsl:value-of
select="$workspaceName"/></gco:CharacterString>
        </name>
        <description>
          <gco:CharacterString><xsl:value-of
select="$workspaceDescr"/></gco:CharacterString>
        </description>
        <function>
          <CI_OnLineFunctionCode
            codeList="{ $codelist}#CI_OnLineFunctionCode"
            codeListValue="download">download</CI_OnLineFunctionCode>
          </function>
        </CI_OnlineResource>
      </onLine>
    </xsl:template>

<xsl:template name="distributionOnline">
  <xsl:call-template name="webmdsEndpointOnline"/>
  <xsl:choose>

```



```
<!-- entry_id = 2101677 is an exception -->
  <xsl:when test="$entryAcro='ORCA05-KAB042_COREforcing'">
    <xsl:call-template name="Online-IFM-GEOMAR-DODS"/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:call-template name="Online-DKRZ"/>
  </xsl:otherwise>
</xsl:choose>
<!--
-->
</xsl:template>

<xsl:template name="contentInfoData">
  <!-- z.B. Variablen im Datensatz -->
  <xsl:for-each select="/CERA2/TOPIC_unit/ROWSET/ROW">
    <xsl:variable name="codeVar">
      <xsl:value-of select="CODE_NUMBER"/>,<xsl:value-of
select="CODE_ACRONYM"/>
    </xsl:variable>
    <xsl:variable name="varCera">
      <xsl:value-of select="TOPIC_NAME"/>
    </xsl:variable>
    <xsl:variable name="paramType">
      <xsl:choose>
        <xsl:when test="UNIT_NAME">physicalMeasurement</xsl:when>
        <xsl:otherwise>thematicClassification</xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <contentInfo>
      <MD_CoverageDescription>
        <attributeDescription>
          <gco:RecordType>
            <xsl:choose>
              <xsl:when test="contains($varCera,$n_f)">
                <xsl:value-of select="$codeVar"/>
              </xsl:when>
              <xsl:otherwise>
                <xsl:value-of select="$varCera"/>
              </xsl:otherwise>
            </xsl:choose>
            <xsl:if test="not
(contains(TOPIC_DESCR,$isCF)=true())">
              <xsl:value-of
select="$delimiter"/><xsl:value-of select="$nonCF"/>
            </xsl:if>
          </gco:RecordType>
        </attributeDescription>
        <contentType>
          <MD_CoverageContentTypeCode
codeList="{ $codelist }#MD_CoverageContentTypeCode"
codeListValue="{ $paramType }"><xsl:val
ue-of select="$paramType"/></MD_CoverageContentTypeCode>
        </contentType>
        <xsl:variable name="vertCRS">
```



```
<xsl:value-of select="UNIT_ACRONYM"/>
</xsl:variable>
<xsl:variable name="vertID">
  <xsl:value-of select="MIN_ALT_UNIT_ACRONYM"/>
</xsl:variable>
<dimension xlink:href="#{$verticalElementPrefix}{$vertID}">
  <MD_RangeDimension>
    <descriptor>
      <gco:CharacterString>
        <xsl:value-of select="UNIT_ACRONYM"/>
      </gco:CharacterString>
    </descriptor>
  </MD_RangeDimension>
</dimension>
</MD_CoverageDescription>
</contentInfo>
</xsl:for-each>
</xsl:template>

<xsl:template name="contentInfo">
<!--
      CC12<xsl:for-each select="/CERA2/TOPIC_ALL_unit_RANGE/ROWSET/ROW">
        <xsl:value-of select="alt_min"/>
      </xsl:for-each>DD12
-->
  <xsl:choose>

    <!-- Flat File projects contains 'DKRZ' but no ocean data -->
<!--
      "contains($dataType, 'DKRZ') and contains($expSummary, 'OPA')='false' and
contains($expSummary, 'NEMO')='false'"
-->
    <xsl:when test="contains($dataType, 'DKRZ') and not
(contains($expSummary, 'OPA') or contains($expSummary, 'NEMO'))">
<!--
      AA<xsl:value-of select="'here we are'"/>AA
-->
    <xsl:for-each
select="/CERA2/TOPIC_ALL_VERT_SC_content/ROWSET/ROW">
      <xsl:variable name="level">
        <xsl:value-of select="VERT_SC"/>
      </xsl:variable>
      <xsl:variable name="codeVar">
        <xsl:value-of select="CODE_NUMBER"/>, <xsl:value-of
select="CODE_ACRONYM"/>
      </xsl:variable>
      <xsl:variable name="varCera">
        <xsl:value-of select="TOPIC_NAME"/>
      </xsl:variable>
      <xsl:variable name="paramType">
        <xsl:choose>
          <xsl:when test="UNIT_NAME">thematicClassification</xsl:when>
          <xsl:otherwise>physicalMeasurement</xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
    </xsl:for-each>
  </xsl:template>

```



```
</xsl:variable>
<contentInfo>
  <MD_CoverageDescription>
    <attributeDescription>
      <gco:RecordType>
        <xsl:choose>
          <xsl:when test="contains($varCera,$n_f)">
            <xsl:value-of select="$codeVar"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:value-of select="$varCera"/>
          </xsl:otherwise>
        </xsl:choose>
        <xsl:if test="not (contains(TOPIC_DESCR,$isCF)=true())">
          <xsl:value-of select="$delimiter"/><xsl:value-of
select="$nonCF"/>
        </xsl:if>
      </gco:RecordType>
    </attributeDescription>
    <contentType>
      <MD_CoverageContentTypeCode
codeList="{ $codelist }#MD_CoverageContentTypeCode"
codeListValue="{ $paramType }"><xsl
:value-of select="$paramType"/></MD_CoverageContentTypeCode>
    </contentType>
    <xsl:choose>
      <xsl:when test="number($level) &#62; 18">
        <dimension
xlink:href="#{$verticalElementPrefix}L{$level}">
          <MD_RangeDimension>
            <descriptor>
              <gco:CharacterString>
                <xsl:value-of select="UNIT_ACRONYM"/>
              </gco:CharacterString>
            </descriptor>
          </MD_RangeDimension>
        </dimension>
        <dimension xlink:href="#{$verticalElementPrefix}hPa">
          <MD_RangeDimension>
            <descriptor>
              <gco:CharacterString>
                <xsl:value-of select="UNIT_ACRONYM"/>
              </gco:CharacterString>
            </descriptor>
          </MD_RangeDimension>
        </dimension>
      </xsl:when>>
      <xsl:when test="(number($level)=1)">
        <dimension xlink:href="#{$verticalElementPrefix}m">
          <MD_RangeDimension>
            <descriptor>
              <gco:CharacterString>
                <xsl:value-of select="UNIT_ACRONYM"/>
              </gco:CharacterString>
            </descriptor>
          </MD_RangeDimension>
        </dimension>
      </xsl:when>>
    </xsl:choose>
  </MD_CoverageDescription>
</contentInfo>
```



```
        </MD_RangeDimension>
      </dimension>
    </xsl:when>>
    <xsl:otherwise>
      <dimension>
        <MD_RangeDimension>
          <descriptor>
            <gco:CharacterString>
              <xsl:value-of select="UNIT_ACRONYM"/>
            </gco:CharacterString>
          </descriptor>
        </MD_RangeDimension>
      </dimension>
    </xsl:otherwise>
  </xsl:choose>
</MD_CoverageDescription>
</contentInfo>
</xsl:for-each>
<xsl:for-each select="/CERA2/TOPIC_ALL_VERT_CHECK/ROWSET/ROW">
  <xsl:variable name="codeVar">
    <xsl:value-of select="CODE_NUMBER"/>,<xsl:value-of
select="CODE_ACRONYM"/>
  </xsl:variable>
  <xsl:variable name="varCera">
    <xsl:value-of select="TOPIC_NAME"/>
  </xsl:variable>
  <xsl:variable name="paramType">
    <xsl:choose>
      <xsl:when test="UNIT_NAME">thematicClassification</xsl:when>
      <xsl:otherwise>physicalMeasurement</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <contentInfo>
    <MD_CoverageDescription>
      <attributeDescription>
        <gco:RecordType>
          <xsl:choose>
            <xsl:when test="contains($varCera,$n_f)">
              <xsl:value-of select="$codeVar"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="$varCera"/>
            </xsl:otherwise>
          </xsl:choose>
            <xsl:if test="not (contains(TOPIC_DESCR,$isCF)=true())">
              <xsl:value-of select="$delimiter"/><xsl:value-of
select="$nonCF"/>
            </xsl:if>
          </gco:RecordType>
        </attributeDescription>
        <contentType>
          <MD_CoverageContentTypeCode
codeList="{ $codelist }#MD_CoverageContentTypeCode"
          codeListValue="{ $paramType }"><xsl
:value-of select="$paramType"/></MD_CoverageContentTypeCode>
```




```
<MD_CoverageContentTypeCode
codeList="{ $codelist}#MD_CoverageContentTypeCode"

codeListValue="{ $paramType}"><xsl:value-of
select="$paramType"/></MD_CoverageContentTypeCode>
</contentType>
  <xsl:variable name="vertCRS">
<xsl:value-of select="UNIT_ACRONYM"/>
</xsl:variable>
  <xsl:variable name="vertID1">
  <xsl:value-of select="MIN_ALT_UNIT_ACRONYM"/>
</xsl:variable>
  <xsl:variable name="vertID2">
  <xsl:value-of select="MAX_ALT_UNIT_ACRONYM"/>
</xsl:variable>
  <xsl:variable name="vertScale">
  <xsl:value-of
select="/CERA2/TOPIC_ALL_unit_RANGE/ROWSET/ROW[topic_name=$varCera]/alt_min"/>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test="$vertID1='hPa'">
      <dimension
xlink:href="#{$verticalElementPrefix}{$vertID1}">
        <MD_RangeDimension>
          <descriptor>
            <gco:CharacterString>
              <xsl:value-of
select="UNIT_ACRONYM"/>
            </gco:CharacterString>
          </descriptor>
        </MD_RangeDimension>
      </dimension>
    </xsl:when>
    <xsl:when test="$vertID1='m'">
      <xsl:choose>
        <xsl:when test="$MPI-OM='MPI-OM' or ($vertID1
= $vertID2)">
          <dimension
xlink:href="#{$verticalElementPrefix}{$vertID1}">
            <MD_RangeDimension>
              <descriptor>
                <gco:CharacterString>
                  <xsl:value-of
select="UNIT_ACRONYM"/>
                </gco:CharacterString>
              </descriptor>
            </MD_RangeDimension>
          </dimension>
        </xsl:when>
        <xsl:otherwise>
          <dimension>
            <MD_RangeDimension>
              <descriptor>
                <gco:CharacterString>
```



```

                                <xsl:value-of
select="UNIT_ACRONYM"/>
                                </gco:CharacterString>
                                </descriptor>
                                </MD_RangeDimension>
                                </dimension>
                                </xsl:otherwise>
</xsl:choose>
    </xsl:when>
                                <xsl:when test="$vertID1 != $vertID2">
                                <dimension
xlink:href="#"{$verticalElementPrefix}{$vertID2}">
                                <MD_RangeDimension>
                                <descriptor>
                                <gco:CharacterString>
                                <xsl:value-of select="UNIT_ACRONYM"/>
                                </gco:CharacterString>
                                </descriptor>
                                </MD_RangeDimension>
                                </dimension>
                                </xsl:when>
                                <xsl:when test="$vertID1 = $vertID2">
                                <dimension
xlink:href="#"{$verticalElementPrefix}{$vertID1}">
                                <MD_RangeDimension>
                                <descriptor>
                                <gco:CharacterString>
                                <xsl:value-of select="UNIT_ACRONYM"/>
                                </gco:CharacterString>
                                </descriptor>
                                </MD_RangeDimension>
                                </dimension>
                                </xsl:when>
                                <xsl:otherwise>
                                <dimension>
                                <MD_RangeDimension>
                                <descriptor>
                                <gco:CharacterString>
                                <xsl:value-of select="UNIT_ACRONYM"/>
                                </gco:CharacterString>
                                </descriptor>
                                </MD_RangeDimension>
                                </dimension>
                                </xsl:otherwise>
                                </xsl:choose>
                                </MD_CoverageDescription>
                                </contentInfo>
                                </xsl:for-each>
                                </xsl:otherwise>

                                </xsl:choose>

</xsl:template>

```



```
<!--experiment-->
<xsl:template name="dataQualityInfo">
  <DQ_DataQuality>
    <scope>
      <DQ_Scope>
        <level>
          <MD_ScopeCode codeList="{ $codelist }#MD_ScopeCode"
codeListValue="experiment">experiment</MD_ScopeCode>
        </level>
      </DQ_Scope>
    </scope>
    <xsl:call-template name="lineage_new">
      <xsl:with-param name="lineage_new_param"
select="'experiment'"/>
    </xsl:call-template>
  </DQ_DataQuality>
</xsl:template>

<!-- for the first step insert new complete lineage block -->
<xsl:template name="lineage_new">
  <xsl:param name="lineage_new_param"/>
  <xsl:variable name="check_type">
    <xsl:if test="$lineage_new_param='dataset' and $ffname=''">
      <xsl:value-of select="'TRUE'" />
    </xsl:if>
    <xsl:if test="$lineage_new_param='experiment' and
/CERA2/STORAGE_ALL/ROWSET/ROW/STORAGE1 = 'ORACLE'">
      <xsl:value-of select="'TRUE'" />
    </xsl:if>
  </xsl:variable>
  <lineage>
    <LI_Lineage>
      <statement>
        <gco:CharacterString>
          <xsl:value-of select="'Model Run'"/>
          <xsl:if test="$check_type='TRUE'">
            ,<xsl:value-of select="'Data Processing CERA'"/>
          </xsl:if>
        </gco:CharacterString>
      </statement>
      <processStep>
        <LI_ProcessStep id="Step_0">
          <description>
            <gco:CharacterString>
              <xsl:value-of select="'Model Run'" />
            </gco:CharacterString>
          </description>
          <dateTime>
            <gco:DateTime>
              <xsl:value-of select="/CERA2/ENTRY/ROWSET/ROW/CREATION_DATE"
/>
            </gco:DateTime>
          </dateTime>
        </LI_ProcessStep>
      </processStep>
    </lineage>
  </xsl:template>

```



```
<xsl:if test="$check_type='TRUE'">
  <processStep>
    <LI_ProcessStep id="Step_1">
      <description>
        <gco:CharacterString>
          <xsl:value-of select="'Data Processing CERA'" />
        </gco:CharacterString>
      </description>
      <dateTime>
        <gco:DateTime>
          <xsl:value-of
select="/CERA2/ENTRY/ROWSET/ROW/CREATION_DATE" />
        </gco:DateTime>
      </dateTime>
    </LI_ProcessStep>
  </processStep>
</xsl:if>
</LI_Lineage>
</lineage>
</xsl:template>

<!--dataset-->
<xsl:template name="dataQualityInfoData">
  <DQ_DataQuality>
    <scope>
      <DQ_Scope>
        <level>
          <MD_ScopeCode codeList="{ $codelist }#MD_ScopeCode"
codeListValue="dataset">dataset</MD_ScopeCode>
        </level>
      </DQ_Scope>
    </scope>
    <xsl:call-template name="lineage_new">
      <xsl:with-param name="lineage_new_param"
select="'dataset'"/>
    </xsl:call-template>
  </DQ_DataQuality>
</xsl:template>

<xsl:template name="substring-after-last-occur">
  <xsl:param name="str"/>
  <xsl:param name="chr"/>
  <xsl:variable name="rest" select="substring-after($str,$chr)" />
  <xsl:variable name="mychr" select="$chr" />
  <xsl:if test="contains($rest,$chr)">
    <xsl:call-template name="substring-after-last-occur">
      <xsl:with-param name="str" select="$rest"/>
      <xsl:with-param name="chr" select="$mychr"/>
    </xsl:call-template>
  </xsl:if>
  <xsl:if test="not(contains($rest,$chr))">
    <xsl:value-of select="$rest"/>
  </xsl:if>
</xsl:template>
```



```
</xsl:template>

<xsl:template name="spatialRepresentationInfo">
  <!-- General information about spatial and time resolution of datasets of
  experiment: Dimensions LON/LAT/ALT/SPEC/TIME !
  In 0.x ter Version des C3Grid Projektes noch nicht auszuf?llen
  spaetere Verwendung !?
  MS: Hier nur Anzahl von Gitterpunkten oder Zeitschritten
  moeglich. Wo spezielle Aufloesung, also z.behandelten Bsp. T63N48L31 200a?
  longitude:N*4/latitude:N*2/altitude:L/spectral:63/time:6h
  Oder eine zusaetzliche spektrale "Achse" einfuehren?
  Wo Aufloesung und deren Einheiten: also 2Grad oder 6h?
  bzw. hPa oder m?
  -->
  <MD_GridSpatialRepresentation>
    <numberOfDimensions>
      <gco:Integer>4</gco:Integer>
    </numberOfDimensions>
    <!-- blocks for each dimension: here for example
    LON/LAT/SPECTRAL/ALT/TIME -->
    <!-- LON -->
    <axisDimensionProperties>
      <MD_Dimension>
        <dimensionName>
          <MD_DimensionNameTypeCode codeListValue="row"
          codeList="{ $codelist }#MD_DimensionNameT
          ypeCode">row</MD_DimensionNameTypeCode>
        </dimensionName>
        <dimensionSize>
          <gco:Integer>192</gco:Integer>
        </dimensionSize>
        <!-- MS: GML UOM=UnitOfMeasure definition usable?
        aus ISO19103, ISO19111: <resolution> <Length, Distance, and Scale
        mit attribut uom>
        -->
        <resolution>
          <gco:Distance uom="2.0">2.0</gco:Distance>
          <!--
          <gco:Scale
          uom="degree">degree</gco:Scale> -->
        </resolution>
      </MD_Dimension>
    </axisDimensionProperties>
    <!-- LAT -->
    <axisDimensionProperties>
      <MD_Dimension>
        <dimensionName>
          <MD_DimensionNameTypeCode
          codeList="{ $codelist }#MD_DimensionNameTypeCode"
          codeListValue="column">column</MD_DimensionNameTypeCode>
        </dimensionName>
        <dimensionSize>
          <gco:Integer>96</gco:Integer>
        </dimensionSize>
        <!-- MS: GML UOM=UnitOfMeasure definition usable???
```



```

    aus ISO19103, ISO19111: <resolution> <Length, Distance, and
Scale mit attribut uom>
-->
<resolution>
  <gco:Distance uom="2.0">2.0</gco:Distance>
  <!-- <gco:Scale uom="degree">degree</gco:Scale> -->
</resolution>
</MD_Dimension>
</axisDimensionProperties>
<!-- SPECTRAL -->
<axisDimensionProperties>
  <MD_Dimension>
    <dimensionName>
      <MD_DimensionNameTypeCode codeListValue="track"
codeList="{ $codelist }#MD_DimensionNameT
ypeCode">track</MD_DimensionNameTypeCode>
    </dimensionName>
    <dimensionSize>
      <gco:Integer>63</gco:Integer>
    </dimensionSize>
  </MD_Dimension>
</axisDimensionProperties>
<!-- ALT -->
<axisDimensionProperties>
  <MD_Dimension>
    <dimensionName>
      <MD_DimensionNameTypeCode codeListValue="vertical"
codeList="{ $codelist }#MD_DimensionNameT
ypeCode">vertical</MD_DimensionNameTypeCode>
    </dimensionName>
    <dimensionSize>
      <gco:Integer>31</gco:Integer>
    </dimensionSize>
  </MD_Dimension>
</axisDimensionProperties>
<!-- TIME -->
<axisDimensionProperties>
  <MD_Dimension>
    <dimensionName>
      <MD_DimensionNameTypeCode
codeList="{ $codelist }#MD_DimensionNameTypeCode"
codeListValue="time">time</MD_DimensionNameTypeCode>
    </dimensionName>
    <dimensionSize>
      <gco:Integer>1752000</gco:Integer>
    </dimensionSize>
    <!-- MS: GML UOM=UnitOfMeasure definition usable???
```

aus ISO19103, ISO19111: <resolution> <Length, Distance, and Scale mit attribut uom> -->

```

    <resolution>
      <gco:Distance uom="6.0">6.0</gco:Distance>
      <!--
-->
                                <gco:Scale uom="hours">hours</gco:Scale>
-->
    </resolution>
  </MD_Dimension>

```



```
</axisDimensionProperties>
<cellGeometry>
  <MD_CellGeometryCode codeListValue="area"
                        codeList="{ $codelist }#MD_CellGeometryCode">area<
/MD_CellGeometryCode>
</cellGeometry>
<transformationParameterAvailability>
  <gco:Boolean>0</gco:Boolean>
</transformationParameterAvailability>
</MD_GridSpatialRepresentation>
</xsl:template>

<xsl:template name="referenceSystemInfo">
  <!-- Info zu verwendetem Koordinatensystem...
       referenceSystemInfo aus 19111 CRS: Coordinate Reference System
       Kresse S.81; Ellipsoidal CRS or Cartesian CRS
       moeglich, aber nicht in 19139!
  -->
  <MD_ReferenceSystem>
    <referenceSystemIdentifier>
      <RS_Identifier>
        <code>
          <gco:CharacterString>CRS:Cartesian</gco:CharacterString>
        </code>
      </RS_Identifier>
    </referenceSystemIdentifier>
  </MD_ReferenceSystem>
</xsl:template>

<!-- Template header
<xsl:template name="gen_header">
  <xsl:comment>** Generated from internal WDCC metadata schema
**</xsl:comment>
  <xsl:comment>** by cera_map_iso.xsql/cera_map_iso.xsl **</xsl:comment>
  <xsl:comment>** (Rev. <xsl:value-of select="$rev"/>, 2006)
**</xsl:comment>
  <xsl:comment>** Version of XSLT Processor: <xsl:value-of select="system-
property('xsl:version')"/>**</xsl:comment>
  <xsl:comment>** XSLT Processor: <xsl:value-of select="system-
property('xsl:vendor')"/> **</xsl:comment>
  <xsl:comment>** URL: <xsl:value-of select="system-property('xsl:vendor-
url')"/> **</xsl:comment>
</xsl:template>
  <MD_MyTest>
    <xsl:value-of select="$whoami"/>
  </MD_MyTest>
  <MyTest><xsl:value-of select="'aaa'"/></MyTest>

  -->

<xsl:template name="iso_header">
  <xsl:comment>*** Generated from internal WDCC metadata schema by
'cera_map_iso.xsql/cera_map_iso.xsl' ***</xsl:comment>
```



```
<xsl:comment>*** C3Grid Template URL is: '<xsl:value-of
select="$metadataTemplateURL"/>' ***</xsl:comment>
</xsl:template>

<!-- MAIN -->
<xsl:template match="/">
  <xsl:variable name="whoami">
    <xsl:value-of select="/CERA2/ENTRY/ROWSET/ROW/ENTRY_TYPE"/>
  </xsl:variable>
  <!--experiment
<xsl:call-template name="iso_header"/>
  -->
  <xsl:choose>
    <xsl:when test="$whoami='experiment'">
      <xsl:call-template name="cera_map_iso_exp"/>
    </xsl:when>
    <xsl:when test="$whoami='dataset'">
      <xsl:call-template name="cera_map_iso_ds"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="cera_map_iso_exp"/>
    </xsl:otherwise>
  </xsl:choose>

  <!--experiment
  <xsl:for-each select="/CERA2/CONNE/ROWSET/ROW">
<xsl:if test="position()=1">
  <xsl:call-template name="cera_map_iso_exp"/>
</xsl:if>
</xsl:for-each>
  -->
  <!--dataset
  <xsl:for-each select="/CERA2/CONNE_DS/ROWSET/ROW">
<xsl:if test="position()=1">
  <xsl:call-template name="cera_map_iso_ds"/>
</xsl:if>
</xsl:for-each>
  -->
</xsl:template>
</xsl:stylesheet>
```



e) XSQL Oracle database access script

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cera_map_iso.xsl"?>
<CERA2
  connection="cera2"
  xmlns:xsql="urn:oracle-xsql"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsql:set-stylesheet-param name="request" value="{@request}" ignore-empty-
value="yes" />
  <xsql:set-stylesheet-param name="metadataPrefix" value="{@metadataPrefix}"
ignore-empty-value="yes" />
  <xsql:set-stylesheet-param name="ffname" value="{@ffname}" ignore-empty-
value="yes" />
  <xsql:set-stylesheet-param name="debug" value="{@debug}" ignore-empty-
value="yes" />

  <!-- ID of dataset to select -->
  <xsql:set-page-param name="Pid" value="{@id}"/>

  <xsql:set-page-param name="Pdsname" value="{@dsname}"/>

  <!-- ID of our own institute
  <xsql:set-page-param name="inst_id" value="1"/>
  -->

  <!-- REF_TYPE_ID=2000004 results in DOI experiments only -->
  <xsql:set-page-param name="doi_exp_id" value="2000004"/>

  <!-- REF_TYPE_ID=2000006 results in datasets only -->
  <xsql:set-page-param name="doi_data_id" value="2000006"/>

  <!-- WDC DOI_PREFIX -->
  <xsql:set-page-param name="doi_prefix" value="'10.1594/WDC/'"/>

  <!-- new query: IS_DOI_EXP ===== -->
  <IS_DOI_EXP>
  <xsql:query>
    select
      access_spec as DOI_STR
    from CERA2.FLAT_REFER
    where REF_TYPE_ID={@doi_exp_id}
    and entry_id = {@Pid}
  </xsql:query>
  </IS_DOI_EXP>

  <!-- new query: IS_DOI_DS ===== -->
  <IS_DOI_DS>
  <xsql:query>
    select
      access_spec as DOI_STR
    from CERA2.FLAT_REFER
    where REF_TYPE_ID={@doi_data_id}
```



```
    and entry_id = {@Pid}
</xsql:query>
</IS_DOI_DS>

    <!-- new query: ACT_DATETIME =====> -->
<ACT_DATETIME>
<xsql:query>
    select distinct
        concat(concat(TO_CHAR( SYSDATE, 'YYYY-MM-DD' ), 'T'), TO_CHAR(
SYSDATE, 'HH24:MI:SS' )) as ACT_DATETIME,
        concat(concat(TO_CHAR( SYSDATE, 'YYYY-MM-DD' ), 'T'), TO_CHAR(
SYSDATE, 'HH24:MI' )) as ACT_DATETIME2,
        TO_CHAR( SYSDATE, 'HH24:MI:SS' ) as ACT_TIME,
        TO_CHAR( SYSDATE, 'YYYY-MM-DD' ) as ACT_DATE
    from cera2.flat_entry
    where entry_id = {@Pid}
</xsql:query>
</ACT_DATETIME>

<!-- new query: SUM_STEP_COUNT =====> -->
<SUM_STEP_COUNT>
<xsql:query>
    select
        sum(target_step_count) as TOT_STEP_COUNT
    from cera2.external_pointer
    where entry_id in (select id_s from cera2.flat_conne where id_g =
{@Pid})
</xsql:query>
</SUM_STEP_COUNT>

    <!-- new query: CAMPA =====> -->
<CAMPA>
<xsql:query>
    select
        ENTRY_ID,
        PROJECT_ACRONYM,
        PROJECT_NAME,
        PROJECT_DESCR
    from CERA2.FLAT_CAMPA
    where ENTRY_ID={@Pid}
</xsql:query>
</CAMPA>

    <!-- new query: CAMPA_ALL =====> -->
<CAMPA_ALL>
<xsql:query>
    select
        ENTRY_ID,
        PROJECT_ACRONYM,
        PROJECT_NAME,
        PROJECT_DESCR
```



```

    from CERA2.FLAT_CAMPA
</xsql:query>
</CAMPA_ALL>

    <!-- new query: CHECK_ ===== -->
<CHECK_EXT_DS>
<xsql:query>
    select
        count(distinct entry_id) as cnt
    from cera2.external_pointer
    where ENTRY_ID = {@Pid}
</xsql:query>
</CHECK_EXT_DS>

<CHECK_EXT_EXP>
<xsql:query>
    select
        count(distinct entry_id) as cnt
    from cera2.external_pointer
    where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G =
{@Pid})
</xsql:query>
</CHECK_EXT_EXP>

<CHECK_CERA_EXP>
<xsql:query>
    select
        count(distinct ID_S) as cnt
    from cera2.flat_conne
    where ID_G = {@Pid}
</xsql:query>
</CHECK_CERA_EXP>

<CHECK_CERA_DS>
<xsql:query>
    select
        count(distinct ID_G) as cnt
    from cera2.flat_conne
    where ID_S = {@Pid}
</xsql:query>
</CHECK_CERA_DS>

<!-- new query: CITATION ===== -->
<CITATION>
<xsql:query>
    SELECT
        concat(concat(concat(concat(TITLE, ' '), FIRST_NAME), '
'), LAST_NAME) as NAME,
        INSTITUTE_NAME,
        CONTACT_TYPE,
        INST_URL
    from cera2.v_citation_contact
    where ENTRY_ID = {@Pid}

```



```
</xsql:query>  
</CITATION>
```

```
<!-- new query: CONNE ===== -->  
<!-- actual not used  
-->
```

```
<CONNE>  
<xsql:query>  
  select  
    ID_G,  
    ENTRY_TYPE_G,  
    ID_S,  
    ENTRY_TYPE_S,  
    NAME_G,  
    ACRO_G  
  from CERA2.FLAT_CONNE  
  where ID_G={@Pid}  
</xsql:query>  
</CONNE>
```

```
<!-- new query: CONNE_DS ===== -->
```

```
<CONNE_DS>  
<xsql:query>  
  select distinct  
    ID_G  
  from CERA2.FLAT_CONNE  
  where ID_S = {@Pid}  
</xsql:query>  
</CONNE_DS>
```

```
<!-- new query: CONTA ===== -->
```

```
<CONTA>  
<xsql:query>  
  select  
    ENTRY_ID,  
    CONTACT_TYPE,  
    CONTACT_TYPE_ID,  
    CONTACT_TYPE_DESCR,  
    INSTITUTE_ID,  
    INSTITUTE_INFO,  
    INSTITUTE_NAME,  
    INSTITUTE_ACRONYM,  
    DEPARTMENT_NAME,  
    DEPARTMENT_ACRONYM,  
    COUNTRY,  
    STATE_OR_PROVINCE,  
    PLACE,  
    STREET,  
    STREET_POSTAL_CODE,  
    POBOX,  
    POBOX_POSTAL_CODE,  
    INST_URL,  
    ADDITIONAL_INFO,
```



```

    PINSTITUTE_INFO,
    PINSTITUTE_NAME,
    PINSTITUTE_ACRONYM,
    PDEPARTMENT_NAME,
    PDEPARTMENT_ACRONYM,
    PCOUNTRY,
    PSTATE_OR_PROVINCE,
    PPLACE,
    PSTREET,
    PSTREET_POSTAL_CODE,
    PPOBOX,
    PPOBOX_POSTAL_CODE,
    PINST_URL,
    PADDITIONAL_INFO,
    PERSON_NAME,
    FIRST_NAME,
    SECOND_NAME,
    LAST_NAME,
    TITLE,
    PERS_URL,
    TELEPHONE,
    EMAIL
  from CERA2.FLAT_CONTA
  where ENTRY_ID={@Pid}
  order by LAST_NAME, FIRST_NAME
</xsql:query>
</CONTA>

```

```

<!-- new query: COVER =====>
<!--

```

```

to_char(to_date(concat(concat(concat(concat(to_char(START_YEAR,'9999'),to_
char(START_MONTH,'99')),to_char(START_DAY,'99')),to_char(0,'99')),to_
char(0,'99'))),'YYYYMMDDHH24MI'),'YYYY-MM-DD"T"HH24:MI:SS') START_DATE,

```

```

to_char(to_date(concat(concat(concat(concat(to_char(STOP_YEAR,'9999'),to_
char(STOP_MONTH,'99')),to_char(STOP_DAY,'99')),to_char(0,'99')),to_
char(0,'99'))),'YYYYMMDDHH24MI'),'YYYY-MM-DD"T"HH24:MI:SS') STOP_DATE

```

```

-->
<COVER>
<xsql:query>
  select
    CURRENTNESS_REF_DESCR,
    MIN_LAT,
    MAX_LAT,
    MIN_LON,
    MAX_LON,
    MIN_ALTITUDE,
    MAX_ALTITUDE,
    MIN_ALT_UNIT_ACRONYM,
    MAX_ALT_UNIT_ACRONYM,
    START_YEAR,
    START_MONTH,

```



```
START_DAY,
STOP_YEAR,
STOP_MONTH,
STOP_DAY
from CERA2.FLAT_COVER
where ENTRY_ID={@Pid}
</xsql:query>
</COVER>

<COVER_hPa>
<xsql:query>
select distinct
min(MIN_ALTITUDE) as MINIMUM_ALT,
max(MAX_ALTITUDE) as MAXIMUM_ALT
from CERA2.FLAT_COVER
where ENTRY_ID = {@Pid}
and MIN_ALT_UNIT_ACRONYM = 'hPa'
</xsql:query>
</COVER_hPa>

<COVER_m>
<xsql:query>
select distinct
min(MIN_ALTITUDE) as MINIMUM_ALT,
max(MAX_ALTITUDE) as MAXIMUM_ALT
from CERA2.FLAT_COVER
where ENTRY_ID = {@Pid}
and MIN_ALT_UNIT_ACRONYM = 'm'
</xsql:query>
</COVER_m>

<COVER_ALL>
<xsql:query>
select distinct
MIN_LAT,
MAX_LAT,
MIN_LON,
MAX_LON,
MIN_ALTITUDE,
MAX_ALTITUDE,
MIN_ALT_UNIT_ACRONYM,
MAX_ALT_UNIT_ACRONYM
from CERA2.FLAT_COVER
where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G = {@Pid})
</xsql:query>
</COVER_ALL>

<COVER_ALL_hPa>
<xsql:query>
select distinct
min(MIN_ALTITUDE) as MINIMUM_ALT,
max(MAX_ALTITUDE) as MAXIMUM_ALT
from CERA2.FLAT_COVER
```



```

        where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G =
{@Pid})
        and (MIN_ALT_UNIT_ACRONYM = 'hPa' or MAX_ALT_UNIT_ACRONYM = 'hPa')
</xsql:query>
</COVER_ALL_hPa>

<COVER_ALL_m>
<xsql:query>
    select distinct
        min(MIN_ALTITUDE) as MINIMUM_ALT,
        max(MAX_ALTITUDE) as MAXIMUM_ALT
    from CERA2.FLAT_COVER
    where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G =
{@Pid})
        and (MIN_ALT_UNIT_ACRONYM = 'm' or MAX_ALT_UNIT_ACRONYM = 'm')
</xsql:query>
</COVER_ALL_m>

<!-- new query: DISTR ===== -->
<DISTR>
<xsql:query>
    select
        ACCESS_CONSTRAINT_DESCR,
        USE_CONSTRAINT_DESCR
    from CERA2.FLAT_DISTR
    where ENTRY_ID = {@Pid}
    order by ACCESS_CONSTRAINT_DESCR, USE_CONSTRAINT_DESCR
</xsql:query>
</DISTR>

<DISTR_ALL>
<xsql:query>
    select distinct
        ACCESS_CONSTRAINT_DESCR,
        USE_CONSTRAINT_DESCR
    from CERA2.FLAT_DISTR
    where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G =
{@Pid})
    order by ACCESS_CONSTRAINT_DESCR, USE_CONSTRAINT_DESCR
</xsql:query>
</DISTR_ALL>

<!-- new query: ENTRY_xxx ===== -->
<ENTRY>
<xsql:query>
    select
        ENTRY_ID,
        ENTRY_NAME,
        ENTRY_ACRONYM,
        TO_CHAR( CREATION_DATE,      'YYYY-MM-DD"T"HH24:MI:SS' ) CREATION_DATE,
        TO_CHAR( REVIEW_DATE,       'YYYY-MM-DD"T"HH24:MI:SS' ) REVIEW_DATE,
        TO_CHAR( FROM_TZ(CAST(REVIEW_DATE as TIMESTAMP), 'EUROPE/Berlin') at
time zone 'GMT'

```



```

, 'YYYY-MM-DD"T"HH24:MI:SS' )
REVIEW_DATE_OAI,
  TO_CHAR( FUTURE_REVIEW_DATE, 'YYYY-MM-DD"T"HH24:MI:SS' )
FUTURE_REVIEW_DATE,
  PROGRESS_DESCR,
  PROGRESS_ACRONYM,
  SUMMARY,
  ENTRY_TYPE,
  ACCURACY_REPORT,
  CONSISTENCY_REPORT,
  HORIZONTAL_ACC_REPORT,
  TO_CHAR( NOW, 'YYYY-MM-DD HH24:MI:SS' )
NOW
  from
  CERA2.FLAT_ENTRY
  where
  ENTRY_ID={@Pid}
</xsql:query>
</ENTRY>

```

```

<!-- trim(replace(to_char(target_size, '999,999,999,999,999,999'), ',', '.'))
target_size,

```

```

concat(to_char(to_date(concat(concat(concat(concat(to_char(m1.year, '9999'), to
_char(m1.month, '99')), to_char(m1.day, '99')), to_char(m1.hour, '99')), to_char(m1
.minute, '99')), 'YYYYMMDDHH24MI'), 'YYYY-MM-DD"T"HH24:MI:SS'), 'Z') START_DATE,

```

```

concat(concat(concat(concat(concat(concat(concat(concat(concat(concat(concat(
concat('P', to_char(s1.year, '9999')), 'Y'), to_char(s1.month, '99')), 'M'), to_char
(s1.day, '99')), 'DT'), to_char(s1.hour, '99')), 'H'), to_char(s1.minute, '99')), 'M'
), to_char(s1.second, '99')), 'S') STEP_DATE_STR

```

-->

```

<ENTRY_ALL_EXT>
<xsql:query>
  select
    target,
    round(target_size / 1000000) as DATASET_SIZE,
    target_step_count,
    m1.year as START_YEAR,
    m1.month as START_MONTH,
    m1.day as START_DAY,
    m1.hour as START_HOUR,
    m1.minute as START_MIN,
    s1.year as STEP_YEAR,
    s1.month as STEP_MONTH,
    s1.day as STEP_DAY,
    s1.hour as STEP_HOUR,
    s1.minute as STEP_MIN,
    start_moment_id,
    step_moment_id,

```



```
concat (concat (concat (concat (concat (concat (concat (concat (concat (concat (
concat ('P',to_char(s1.year *
target_step_count)), 'Y'),to_char(s1.month, '99')), 'M'),to_char(s1.day *
target_step_count)), 'DT'),to_char(s1.hour *
target_step_count)), 'H'),to_char(s1.minute *
target_step_count)), 'M'),to_char(s1.second * target_step_count)), 'S')
STEP_DATE_STR
from
    cera2.external_pointer,
    cera2.moment m1,
    cera2.moment s1
where
    m1.moment_id = cera2.external_pointer.start_moment_id
    and s1.moment_id = cera2.external_pointer.step_moment_id
    and entry_id = {@Pid}
    and target like '%{@ffname}'
</xsql:query>
</ENTRY_ALL_EXT>
```

```
<ENTRY_EXP_SUM>
<xsql:query>
    select distinct
        SUMMARY
    from CERA2.FLAT_ENTRY
    where ENTRY_ID = (select ID_G from CERA2.flat_conne where ID_S = {@Pid})
</xsql:query>
</ENTRY_EXP_SUM>
```

<!-- new query: EXP_xxx =====>

```
concat (concat (concat (concat (concat (concat (concat (concat (concat (concat (
concat ('P',to_char(s1.year, '9999')), 'Y'),to_char(s1.month, '99')), 'M'),to_char
(s1.day, '99')), 'DT'),to_char(s1.hour, '99')), 'H'),to_char(s1.minute, '99')), 'M'
),to_char(s1.second, '99')), 'S') END_DATE_STR
-->
<EXP_SIZE>
<xsql:query>
    select
        round(DATA_SIZE / 1000000) as TOTAL_SIZE
    from CERA2.v_experiment_datasize
    where experiment_id = {@Pid}
</xsql:query>
</EXP_SIZE>

<EXT_STEP>
<xsql:query>
    select distinct
        target,
        round(target_size / 1000000) as DATASET_SIZE,
        target_step_count,
        STEP_YEAR,
```



```
        STEP_MONTH,
        STEP_DAY,
        STEP_HOUR,
        STEP_MINUTE
    from cera2.v_external_moment
    where entry_id in (select ID_S from CERA2.flat_conne where ID_G =
{@Pid})
    order by target
</xsql:query>
</EXT_STEP>

<!-- new query: EXPID ===== -->
<EXPID>
<xsql:query>
    select ID_G
        from CERA2.flat_conne
    where ID_S = {@Pid}
</xsql:query>
</EXPID>

<!-- new query: FORMAT ===== -->
<FORMAT>
<xsql:query>
select distinct
    FORMAT_ACRONYM,
    FORMAT_DESCR
from
    CERA2.FLAT_DISTR
where
    ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G = {@Pid})
</xsql:query>
</FORMAT>

<FORMAT_DS>
<xsql:query>
    select distinct
        FORMAT_ACRONYM,
        FORMAT_DESCR,
        round(DATA_SIZE / 1000000) as DATASET_SIZE
    from
        CERA2.FLAT_DISTR
    where
        ENTRY_ID = {@Pid}
</xsql:query>
</FORMAT_DS>

<!-- new query: KEY_C ===== -->
<KEY_C>
<xsql:query>
    select
        ENTRY_ID,
        GENERAL_KEY
    from CERA2.FLAT_KEY_C
```



```

        where ENTRY_ID={@Pid}
    </xsql:query>
</KEY_C>

<KEY_C_EXP>
<xsql:query>
    select
        GENERAL_KEY
    from CERA2.FLAT_KEY_C
    where ENTRY_ID in (select ID_G from CERA2.flat_conne where ID_S =
{@Pid})
    </xsql:query>
</KEY_C_EXP>

<!-- new query: MOMENT ===== -->
<MOMENT>
<xsql:query>
    select distinct
        NUM_POINTS,
        START_HOUR,
        START_MINUTE,
        START_SECOND,
        STEP_YEAR,
        STEP_MONTH,
        STEP_DAY,
        STEP_HOUR,
        START_YEAR,
        START_MONTH,
        START_DAY
    from CERA2.FLAT_PARAM, cera2.DATA_ORG, cera2.V_EQUIDISTANT_TIME
    where cera2.flat_param.ENTRY_ID in (select ID_S from CERA2.flat_conne
where ID_G = {@Pid})
    and cera2.flat_param.DATA_ORG_ID=cera2.DATA_ORG.data_org_id
    and cera2.DATA_ORG.time_id = cera2.V_EQUIDISTANT_TIME.time_id
    and (STEP_YEAR != 0 OR STEP_MONTH != 0 OR STEP_DAY != 0 OR STEP_HOUR != 0)
    </xsql:query>
</MOMENT>
<!-- old version:
    from cera2.V_EQUIDISTANT_TIME, CERA2.FLAT_PARAM, cera2.DATA_ORG
    where cera2.DATA_ORG.data_org_id = cera2.flat_param.DATA_ORG_ID
    and cera2.DATA_ORG.time_id = cera2.V_EQUIDISTANT_TIME.time_id
    and cera2.flat_param.ENTRY_ID in (select ID_S from CERA2.flat_conne where
ID_G = {@Pid})
    and (STEP_YEAR != 0 OR STEP_MONTH != 0 OR STEP_DAY != 0 OR STEP_HOUR != 0)

to_char(to_date(concat(concat(concat(concat(to_char(START_YEAR,'9999')),to_cha
r(START_MONTH,'99')),to_char(START_DAY,'99')),to_char(START_HOUR,'99')),to_ch
ar(START_MINUTE,'99')), 'YYYYMMDDHH24MI'), 'YYYY-MM-DD"T"HH24:MI:SS')
START_DATE
-->

<!-- new query: REFER ===== -->
<!-- actual not used
-->

```



```
<REFER>
<xsql:query>
  select
    ENTRY_ID,
    REF_TYPE_ID,
    REF_TYPE_DESCR,
    TITLE,
    AUTHORS,
    PUBLICATION,
    PUBLISHER,
    EDITOR,
    TO_CHAR( PUBLICATION_DATE, 'YYYY-MM-DD' ) PUBLICATION_DATE,
    COUNTRY,
    STATE,
    PLACE,
    EDITION,
    ACCESS_SPEC,
    ADDITIONAL_INFO,
    PRESENTATION_DESCR,
    CITATION_ID,
    CITATION_TYPE,
    CITATION_TYPE_DESCR,
    CITATION_INFO
  from CERA2.FLAT_REFER
  where ENTRY_ID={@Pid}
</xsql:query>
</REFER>
```

```
<REFER_DATA>
<xsql:query>
  select
    ENTRY_ID,
    REF_TYPE_ID,
    REF_TYPE_DESCR,
    TITLE,
    AUTHORS,
    PUBLICATION,
    PUBLISHER,
    EDITOR,
    TO_CHAR( PUBLICATION_DATE, 'YYYY-MM-DD' ) PUBLICATION_DATE,
    COUNTRY,
    STATE,
    PLACE,
    EDITION,
    ACCESS_SPEC,
    ADDITIONAL_INFO,
    PRESENTATION_DESCR,
    CITATION_ID,
    CITATION_TYPE,
    CITATION_TYPE_DESCR,
    CITATION_INFO
  from CERA2.FLAT_REFER
  where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G = {@Pid})
</xsql:query>
```



</REFER_DATA>

<!-- new query: SPATI ===== -->

```

<SPATI>
<xsql:query>
  select
    ENTRY_ID,
    HOR_SYS_DESCR,
    VER_SYS_DESCR
  from CERA2.FLAT_SPATI
  where ENTRY_ID={@Pid}
</xsql:query>
</SPATI>

```

<!-- new query: SPATI_ALL ===== -->

```

<SPATI_ALL>
<xsql:query>
  select distinct
    HOR_SYS_DESCR,
    VER_SYS_DESCR
  from CERA2.FLAT_SPATI
  where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G = {@Pid})
</xsql:query>
</SPATI_ALL>

```

<!-- new query: STORAGE ===== -->

```

<STORAGE>
<xsql:query>
  select distinct
    STORAGE1,
    STORAGE2
  from cera2.V_ENTRY_STORAGE
  where ENTRY_ID = {@Pid}
</xsql:query>
</STORAGE>

```

```

<STORAGE_ALL>
<xsql:query>
  select distinct
    STORAGE1
  from cera2.V_ENTRY_STORAGE
  where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G = {@Pid})
</xsql:query>
</STORAGE_ALL>

```

<!-- new query: TIMESTEP ===== -->

```

<TIMESTEP>
<xsql:query>
  select
    STEP_YEAR,
    STEP_MONTH,
    STEP_DAY,

```



```
STEP_HOUR,
STEP_MINUTE,
STEP_SECOND,
```

```
concat (concat (concat (concat (concat (concat (concat (concat (to_char (STEP_Y
EAR, '9999'), 'Y'), to_char (STEP_MONTH, '99')), 'M')
, to_char (STEP_DAY, '99')), 'DT'), to_char (STEP_HOUR, '99')), 'H'), to_char (STEP_MIN
UTE, '99')), 'M') STEP_DATE_STR
```

```
from CERA2.V_ENTRY_EQUIDISTANT_TIME
where ENTRY_ID = {@Pid}
</xsql:query>
</TIMESTEP>
```

```
<!-- new query: TOPIC =====>
```

```
<TOPIC>
<xsql:query>
select distinct
TOPIC_ID,
TOPIC_NAME,
TOPIC_DESCR,
CODE_NUMBER,
CODE_TYPE,
CODE_ACRONYM,
CODE_DESCR,
UNIT_NAME,
UNIT_ACRONYM,
REFERENCE_METHOD
from CERA2.FLAT_PARAM
where ENTRY_ID = {@Pid}
order by TOPIC_NAME
</xsql:query>
</TOPIC>
```

```
-->
```

```
<TOPIC_unit>
<xsql:query>
select distinct
TOPIC_NAME,
cera2.flat_cover.MIN_ALT_UNIT_ACRONYM,
TOPIC_DESCR,
CODE_NUMBER,
CODE_TYPE,
CODE_ACRONYM,
CODE_DESCR,
UNIT_NAME,
UNIT_ACRONYM,
REFERENCE_METHOD
from CERA2.FLAT_PARAM, cera2.flat_cover
where cera2.flat_param.ENTRY_ID = {@Pid}
and cera2.flat_param.entry_id=cera2.flat_cover.entry_id
order by cera2.flat_cover.MIN_ALT_UNIT_ACRONYM,
CERA2.FLAT_PARAM.TOPIC_NAME
</xsql:query>
</TOPIC_unit>
```



```
<!-- new query: TOPIC_ALL ===== -->
<TOPIC_ALL>
<xsql:query>
  select distinct
    TOPIC_ID,
    TOPIC_NAME,
    TOPIC_DESCR,
    CODE_NUMBER,
    CODE_TYPE,
    CODE_ACRONYM,
    CODE_DESCR,
    UNIT_NAME,
    UNIT_ACRONYM,
    REFERENCE_METHOD
  from CERA2.FLAT_PARAM
  where ENTRY_ID in (select ID_S from CERA2.flat_conne where ID_G = {@Pid})
  order by TOPIC_NAME
</xsql:query>
</TOPIC_ALL>

<TOPIC_ALL_unit>
<xsql:query>
  select distinct
    TOPIC_NAME,
    cera2.flat_cover.MIN_ALT_UNIT_ACRONYM,
    cera2.flat_cover.MAX_ALT_UNIT_ACRONYM,
    TOPIC_DESCR,
    CODE_NUMBER,
    CODE_TYPE,
    CODE_ACRONYM,
    CODE_DESCR,
    UNIT_NAME,
    UNIT_ACRONYM,
    REFERENCE_METHOD
  from CERA2.FLAT_PARAM,cera2.flat_cover
  where cera2.flat_param.ENTRY_ID in (select ID_S from CERA2.flat_conne
where cera.Check_Perm.check_permit_4dsname(CERA2.flat_conne.ACRO_S) = 0 and
ID_G = {@Pid})
  and cera2.flat_param.entry_id=cera2.flat_cover.entry_id
  order by cera2.flat_cover.MIN_ALT_UNIT_ACRONYM,
CERA2.FLAT_PARAM.TOPIC_NAME
</xsql:query>
</TOPIC_ALL_unit>

<TOPIC_ALL_unit_RANGE>
<xsql:query>
  select
    to_char(min(cera2.flat_cover.MIN_ALTITUDE)) as alt_max,
    to_char(max(cera2.flat_cover.MAX_ALTITUDE)) as alt_max
  from cera2.FLAT_COVER,cera2.flat_param
  where cera2.flat_cover.entry_id in (select ID_S from CERA2.flat_conne
where ID_G = {@Pid})
  and cera2.FLAT_COVER.entry_id=cera2.flat_param.entry_id
</xsql:query>
```



```
</TOPIC_ALL_unit_RANGE>

<!--
<TOPIC_ALL_group>
<xsql:query>
    select distinct
        TOPIC_NAME,
            min(MIN_ALTITUDE),
            MIN_ALT_UNIT_ACRONYM,
            max(MAX_ALTITUDE),
            MAX_ALT_UNIT_ACRONYM
        from CERA2.FLAT_COVER, CERA2.FLAT_param,
cera2.V_DATA_ORG_SPACE_3D_GRID
        where CERA2.FLAT_COVER.ENTRY_ID in (select ID_S from
CERA2.flat_conne where ID_G = {@Pid})
        and CERA2.FLAT_param.entry_id=CERA2.FLAT_COVER.ENTRY_ID
        and
cera2.V_DATA_ORG_SPACE_3D_GRID.DATA_ORG_ID=CERA2.FLAT_param.DATA_ORG_ID
        and topic_name in (select distinct topic_name from
CERA2.FLAT_param where CERA2.FLAT_param.ENTRY_ID in (select ID_S from
CERA2.flat_conne where ID_G = {@Pid}))
        group by topic_name,Min_ALT_UNIT_ACRONYM,MAX_ALT_UNIT_ACRONYM
</xsql:query>
</TOPIC_ALL_group>

<TOPIC_ALL_scale_lst>
<xsql:query>
    select distinct
        Z_SCALE,
        TOPIC_NAME,
        MIN_ALTITUDE,
        MIN_ALT_UNIT_ACRONYM,
        MAX_ALTITUDE,
        MAX_ALT_UNIT_ACRONYM
        from CERA2.FLAT_COVER, CERA2.FLAT_param,
cera2.V_DATA_ORG_SPACE_3D_GRID
        where CERA2.FLAT_COVER.ENTRY_ID in (select ID_S from
CERA2.flat_conne where ID_G = {@Pid})
        and CERA2.FLAT_param.entry_id=CERA2.FLAT_COVER.ENTRY_ID
        and
cera2.V_DATA_ORG_SPACE_3D_GRID.DATA_ORG_ID=CERA2.FLAT_param.DATA_ORG_ID
</xsql:query>
</TOPIC_ALL_scale_lst>
-->

<TOPIC_ALL_VERT_SC>
<xsql:query>
    select distinct
        substr(space_acronym,instr(space_acronym,'_L')+2) as VERT_SC
        from CERA2.FLAT_COVER, CERA2.FLAT_param,
cera2.V_DATA_ORG_SPACE_3D_GRID
        where CERA2.FLAT_COVER.ENTRY_ID in (select ID_S from
CERA2.flat_conne where ID_G = {@Pid})
        and CERA2.FLAT_param.entry_id=CERA2.FLAT_COVER.ENTRY_ID
```



```
and
cera2.V_DATA_ORG_SPACE_3D_GRID.DATA_ORG_ID=CERA2.FLAT_param.DATA_ORG_ID
and space_acronym like '%!_L%' escape '!'
</xsql:query>
</TOPIC_ALL_VERT_SC>

<TOPIC_ALL_VERT_SC_content>
<xsql:query>
select distinct
  substr(space_acronym,instr(space_acronym,'_L')+2) as VERT_SC,
  Z_SCALE,
  TOPIC_NAME,
  cera2.flat_cover.MIN_ALT_UNIT_ACRONYM,
  cera2.flat_cover.MAX_ALT_UNIT_ACRONYM,
  TOPIC_DESCR,
  CODE_NUMBER,
  CODE_TYPE,
  CODE_ACRONYM,
  CODE_DESCR,
  UNIT_NAME,
  UNIT_ACRONYM,
  REFERENCE_METHOD
from CERA2.FLAT_COVER, CERA2.FLAT_param, cera2.V_DATA_ORG_SPACE_3D_GRID
where CERA2.FLAT_COVER.ENTRY_ID in (select ID_S from CERA2.flat_conne
where ID_G = {@Pid})
and CERA2.FLAT_param.entry_id=CERA2.FLAT_COVER.ENTRY_ID
and
cera2.V_DATA_ORG_SPACE_3D_GRID.DATA_ORG_ID=CERA2.FLAT_param.DATA_ORG_ID
and space_acronym like '%!_L%' escape '!'
</xsql:query>
</TOPIC_ALL_VERT_SC_content>

<TOPIC_ALL_VERT_CHECK>
<xsql:query>
select distinct
  substr(space_acronym,instr(z_scale,'-')) as VERT_SC,
  TOPIC_NAME,
  cera2.flat_cover.MIN_ALT_UNIT_ACRONYM,
  cera2.flat_cover.MAX_ALT_UNIT_ACRONYM,
  TOPIC_DESCR,
  CODE_NUMBER,
  CODE_TYPE,
  CODE_ACRONYM,
  CODE_DESCR,
  UNIT_NAME,
  UNIT_ACRONYM,
  REFERENCE_METHOD
from CERA2.FLAT_COVER, CERA2.FLAT_param, cera2.V_DATA_ORG_SPACE_3D_GRID
where CERA2.FLAT_COVER.ENTRY_ID in (select ID_S from CERA2.flat_conne
where ID_G = {@Pid})
and CERA2.FLAT_param.entry_id=CERA2.FLAT_COVER.ENTRY_ID
and
cera2.V_DATA_ORG_SPACE_3D_GRID.DATA_ORG_ID=CERA2.FLAT_param.DATA_ORG_ID
and substr(z_scale,1,2) ='ML'
</xsql:query>
```



```

</TOPIC_ALL_VERT_CHECK>

<!--
    SUBSTR(z_scale,1,3) as VERT_SC
    from CERA2.FLAT_COVER, CERA2.FLAT_param, cera2.V_DATA_ORG_SPACE_3D_GRID
    where CERA2.FLAT_COVER.ENTRY_ID in (select ID_S from CERA2.flat_conne
where ID_G = {@Pid})
    and CERA2.FLAT_param.entry_id=CERA2.FLAT_COVER.ENTRY_ID
    and
cera2.V_DATA_ORG_SPACE_3D_GRID.DATA_ORG_ID=CERA2.FLAT_param.DATA_ORG_ID
    and z_scale != 'none'
    and z_scale != 'not filled'

<xsql:ref-cursor-function>
    [SCHEMA_NAME.][PACKAGE_NAME.]FUNCTION_NAME(args);
</xsql:ref-cursor-function>
<xsql:ref-cursor-function>
    cera.Check_Perm.check_permit_4dsname('{@Pdsname}')
</xsql:ref-cursor-function>
    to_char(cera.Check_Perm.check_permit_4dsname('EH5OM_20CSMT_1_6H_ACLCOV'
)) as aaa

<xsql:set-page-param name="datasetLst">
    select distinct
        ACRO_S
    from CERA2.flat_conne
    where ID_G = {@Pid}
</xsql:set-page-param>

<xsql:query>
    select col3,col4
    from table2
    where col3 = {@datasetLst}
</xsql:query>

    <xsql:ref-cursor-function>
</xsql:ref-cursor-function>

<CHECK_EXP_PERM>
<xsql:query>
    select distinct
        cera.Check_Perm.check_permit_4dsname(ACRO_S) as PERM
    from CERA2.flat_conne
    where ID_G = {@Pid}
</xsql:query>
</CHECK_EXP_PERM>
-->

<!-- T h e   E N D   ===== -->

</CERA2>

```