Data Format Standardization of Space Weather Model Output at The Community Coordinated Modeling Center

M. Maddox

NASA Goddard Space Flight Center

http://ccmc.gsfc.nasa.gov

CCMC Workshop
October 11-14, 2005

Tuesday, October 11th PM Session:
Frameworks and Data Infrastructure
What the CCMC provides:

- Scientific validation
- Coupling in collaboration with model owners
- Metrics implementations
- Model runs on request
- Advanced visualization
- **Data Format Standardization**
Covering the Entire Domain
Covering the Entire Domain

- PFSS
- WSA
- Exospheric Solar Wind
- MAS
- Heliospheric Tomography
- ENLIL
- BATSRUS
- Open SGCM
- Fok RC
- Weimer2K
- SAMI2
- CTIP
- SWMF
Challenges

• No rules for standard model interfaces
• Each new model has unique output format
• Developer/user needs to become familiar with internal structure of each output file
• Custom read routines to access model data
• Data typically is not self descriptive
• Reduces portability and reuse of
  – Data output itself
  – Tools created to analyze data
Every Models Output Is Unique

Environment Without Standard

• Specialized I/O routines required for every interface
• Unsuitable for use in flexible model chain
• No commonality between data passing through interfaces
Every Models Output Is Unique

Standardized Environment

- Original output can be preserved
- Standard format for storage, coupling, & visualization
- Model developers continue to have freedom of choice
- Ensures compatibility between models for coupling
- Ground work for which standard, reusable interfaces and tools can be developed

n + m interfaces required
Data Format Standard Options

- CDF
- HDF, HDF4, HDF5
- NetCDF
- FITS
- GRIB
- BUFR
- GRADS
- Office Note 29
- Office Note 84
- VICAR
- PDS
- Open Dx Data Model
Aside from the one-to-one data conversion, what additional **global** & **variable** meta data do we want to provide?

- General description of model, how-to usage – README
- Model name and type
- Date info
  - Run date
  - Generation date
- Grid Description – # of grids, # of dimensions, dimension size(s)
- Coordinate system(s)
- Variable metadata – grid system, min, max, units, description
Conversion Software Components

main conversion routine

main read driver
- read model a routine
- read model b routine
- read model n routine

main write driver
- write model_2_structure
- write structure 2 cdf
- write structure 2 hdf5

registered variables list
- CCMC_name
- native/alias
- x
  - x_pos, xp
- y
  - y_pos, yp

model specific attribute list (.h)
generic/default variable attribute list (.h)
model specific variable attributes list (.h)
generic attribute list (.h)

model specific attribute list (.h)

model specific attribute list (.h)

variable attributes

variable names

variable attributes

structure definitions (.h)

structure manager

structure definitions (.h)

Model Variable List

Registered Variables List

Model Data Assembled Into Standard Data Structures

standard data files with common attributes and variable names for each registered model
Data Access ?
CCMC Access/Interpolation Library

Your Code/Application

CCMC Access/Interpolation Library

Standardized CCMC Model Data

Call from any C supported Programming Language:
- Fortran
- C/C++
- IDL
- OpenDx

- Java
- Perl
- Vtk
- Your App

CCMC Access & Interpolation Library

open_cdf( cdf_name, 0);  get_units( variable_name);
interpolate_batsrus_cdf( variable1, X, Y, Z, 0, 0 );
interpolate_ucla_ggcm_cdf( variable1, X, Y, Z, 0, 0 );
close_cdf();  
gattribute_float_get( attribute_name );
gattribute_char_get( attribute_name );
init_time( data_path, start_time, end_time )
time_interpolate( variable_name, time, X, Y, Z )
vattribute_get( variable_name, attribute_name );

Current Standardized Model Output Availability

BATSRUS
OpenGGCM / UCLA-GGCM
CTIP ( Testing Phase )

Currently Supported Science Data Formats

CDF 2.7
CDF 3.0 ( testing )
HDF5 ( coming soon )
Data Access

- Currently creating a user friendly library of routines that can be called from any C supported language.

<table>
<thead>
<tr>
<th>USER CODE</th>
<th>CCMC Access &amp; Interpolation Library</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>open_cdf( cdf_name, 0);</td>
</tr>
<tr>
<td></td>
<td>interpolate_batsrus_cdf( variable1, X, Y, Z, 0, 0 );</td>
</tr>
<tr>
<td></td>
<td>interpolate_ucla_ggcm_cdf( variable1, X, Y, Z, 0, 0 );</td>
</tr>
<tr>
<td></td>
<td>close_cdf();</td>
</tr>
<tr>
<td></td>
<td>get_units( variable_name);</td>
</tr>
<tr>
<td></td>
<td>gattribute_char_get( attribute_name );</td>
</tr>
<tr>
<td></td>
<td>gattribute_float_get( attribute_name );</td>
</tr>
<tr>
<td></td>
<td>init_time( data_path, start_time, end_time )</td>
</tr>
<tr>
<td></td>
<td>time_interpolate( variable_name, time, X, Y, Z )</td>
</tr>
<tr>
<td></td>
<td>vattribute_get( variable_name, attribute_name );</td>
</tr>
<tr>
<td></td>
<td>FORTRAN INTERFACE</td>
</tr>
</tbody>
</table>
Data Access

• Currently creating a user friendly library of routines that can be called from any C supported language.

```c
open_cdf( cdf_name, 0);
interpolate_batsrus_cdf( variable1, X, Y, Z, 0, 0 );
close_cdf();
```

• Model output can now be easily accessed using CCMC access/interpolation library and standardized files
  – Allows data to be used in any application that supports the standard C programming language
  – Platform independent
General Usage and Benefits

- Speed and efficiency of direct data access
- Self descriptive data files
- Can be used by anyone with CDF tools & libraries
- Same interface regardless of model
- Platform independent
- Promotes data sharing
- Facilitates code reuse
- CCMC access/interpolation libraries allow model data to be easily integrated into any existing analysis software or application
Specific Usage and Benefits

- Allows CCMC to serve high resolution data that would normally be unavailable
- Integrated Access/Interpolation Libraries into CCMC software infrastructure
  - 3D View Visualization Package (IDL) & Runs-On-Request System
  - Space Weather Explorer Visualization Tool (OpenDx)
- Good for metric studies comparing observation data with simulation data
- Several early adopters
  - Visualization of Kelvin-Helmholtz Waves
  - Particle tracing
- In house research & analysis of high resolution runs

Quick & Efficient Data Access

Analysis only required
Vx, Bx, By in
Range of ~10RE

Reduced plot times from 15min to < 5sec
Outstanding Issues

• Variable naming conventions
• How much metadata to pack into each file
• Refining grid description
• Making the transition
• As simulations grow larger keeping original model output may not be feasible
• Add/integrate coordinate system transformation software into existing software suite as well as specific data analysis tools
Summary

• General grid description scheme in place
• Clearly defined set of core metadata elements
• Structure oriented architecture ensures flexibility
• Conversion software currently supports:
  – BATSRUS, BATSRUS SWMF Framework, UCLA-GGCM/OpenGGCM, & CTIP
• CCMC access/interpolation library currently supports:
  – BATSRUS, BATSRUS SWMF Framework, UCLA-GGCM/OpenGGCM CDF files
  – Interface to easily extract global & variable metadata
  – Time interpolation for entire data sets
  – Fortran Interface
• Add Solar & Ionosphere Models to software - CTIP testing phase
• Implement HDF 5 conversion module
• Library is currently available for use and we encourage feedback - questions, comments, and/or suggestions
Data Format Standardization of Space Weather Model Output at The Community Coordinated Modeling Center

M. Maddox

NASA Goddard Space Flight Center

http://ccmc.gsfc.nasa.gov

CCMC Workshop
October 11-14, 2005

Tuesday, October 11th PM Session:
Frameworks and Data infrastructure
Scientific Tool Library

Maxwell's Equations

\[ \oint E \cdot d\mathbf{s} = \frac{Q}{\varepsilon_0} \]

Gauss's Law

\[ \oint B \cdot d\mathbf{l} = 0 \]

(no monopoles)

\[ \oint B \cdot d\mathbf{l} = \mu_0 (\mathbf{J} + \mathbf{D}) \]

Ampère's Law

\[ \oint E \cdot d\mathbf{l} = -\frac{\partial \Phi_B}{\partial t} \]

Faraday's Law

\[ \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \]

\[ \nabla \times \mathbf{B} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \]

\[ \nabla \cdot \mathbf{D} = \rho \]

\[ \nabla \cdot \mathbf{B} = 0 \]

(Differential Forms)
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>README</td>
<td>General description describing the model, contents of CDF, and HOWTO usage.</td>
<td></td>
</tr>
<tr>
<td>README_visualization</td>
<td>Guidelines for visualizing data contained in file.</td>
<td></td>
</tr>
<tr>
<td>model_name</td>
<td>Name of the registered model that produced the data</td>
<td></td>
</tr>
<tr>
<td>model_type</td>
<td>The type of model used to produce data i.e. Global Magnetosphere</td>
<td></td>
</tr>
<tr>
<td>generation_date</td>
<td>Date of generation or run date...</td>
<td></td>
</tr>
<tr>
<td>original_output_file_name</td>
<td>Name of the original model data file that was converted to current CDF file</td>
<td></td>
</tr>
<tr>
<td>run_registration_number</td>
<td>CCMC Runs on Request registration number for runs submitted through online system</td>
<td></td>
</tr>
<tr>
<td>generated_by</td>
<td>Personal identifying info (First Name Last Name)</td>
<td></td>
</tr>
<tr>
<td>terms_of_usage</td>
<td>For tracking purposes for our government sponsors, we ask that you notify the CCMC whenever you use CCMC results in a scientific publication or presentation</td>
<td></td>
</tr>
<tr>
<td>grid_system_count</td>
<td>The number n of how many grid systems are used and/or described in the current cdf file</td>
<td></td>
</tr>
<tr>
<td>grid_system_n_number_of_dimensions</td>
<td>The number m of how many dimensions are in grid n. So for every grid there will be a corresponding grid_system_n_number_of_dimensions attribute i.e. The first grid will have an attribute grid_system_1_number_of_dimensions</td>
<td></td>
</tr>
<tr>
<td>grid_system_n_dimension_m_size</td>
<td>Size of dimension m for grid n</td>
<td></td>
</tr>
<tr>
<td>grid_system_n</td>
<td>Outline how particular grid system is defined by showing coordinates used i.e. [ X,Y,Z ] were X,Y,Z are position variables defined in current CDF</td>
<td></td>
</tr>
<tr>
<td>output_type</td>
<td>Define the type of output is contained in CDF file. i.e. Global Magnetosphere model with Ionosphere output</td>
<td></td>
</tr>
<tr>
<td>standard_grid_target</td>
<td>Defines a standard target grid and coordinate system for which the current models output can be converted to using an external coordinate transformation package</td>
<td></td>
</tr>
<tr>
<td>grid_n_type</td>
<td>Keywords identifying all grids used in current model output. Grid types will be registered in external coordinate transformation package</td>
<td></td>
</tr>
<tr>
<td>start_time</td>
<td>Time in CDF Epoch3 format ( YYYY-MM-DDThh:mm:ss.msecZ ) signifying beginning of the simulation</td>
<td></td>
</tr>
<tr>
<td>end_time</td>
<td>Time in CDF Epoch3 format ( YYYY-MM-DDThh:mm:ss.msecZ ) signifying end of the simulation</td>
<td></td>
</tr>
<tr>
<td>run_type</td>
<td>An event or model extracted from DatabaseInfo file</td>
<td></td>
</tr>
</tbody>
</table>

**CCMC Global Attributes**

- README
- README_visualization
- model_name
- model_type
- Generation_date
- Original_output_file_name
- Run_registration_number
- Generated_by
- Terms_of_usage
- Grid_system_count
- Grid_system_n_number_of_dimensions
- Grid_system_n_dimension_m_size
- Grid_system_n
- Output_type
- Standard_grid_target
- Grid_n_type
- Start_time
- End_time
- Run_type

**Model Specific Global Attributes**

...
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid_min</td>
<td>Smallest valid value for a particular variable. If actual value is less than this valid_min, the actual value is physically impossible and/or was generated in error.</td>
<td></td>
</tr>
<tr>
<td>valid_max</td>
<td>Largest valid value for a particular variable. If actual value is greater than this valid_max, the actual value is physically impossible and/or was generated in error.</td>
<td></td>
</tr>
<tr>
<td>units</td>
<td>The particular units of measurement for a variable</td>
<td></td>
</tr>
<tr>
<td>grid_system</td>
<td>The grid system in which a variable is on</td>
<td></td>
</tr>
<tr>
<td>mask</td>
<td>Mask value</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>A description of the variable</td>
<td></td>
</tr>
<tr>
<td>is_vector_component</td>
<td>Boolean value identifies if variable is a vector component or scalar variable</td>
<td></td>
</tr>
<tr>
<td>position_grid_system</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>data_grid_system</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>actual_min</td>
<td>The smallest value for a particular variable</td>
<td></td>
</tr>
<tr>
<td>actual_max</td>
<td>The largest value for a particular variable</td>
<td></td>
</tr>
</tbody>
</table>
Populating the Structures

- Library of C routines that are used to populate the standard attribute and variable structures.

Model Read Routine

Write Model 2 Structure Routine

CCMC Structure Manager Library

- `init_cmc_attribute_structure`
- `init_cmc_variable_structure`
- `put_ccmc_attribute`
- `update_cmc_attribute_value`
- `update_cmc_variable_value`
- `update_cmc_variable_attribute_value`
- `free_cmc_attribute_structure`
- `free_cmc_variable_structure`

Model Data Assembled Into Standard Data Structures

write structure 2 standard format

standard data files with common attributes and variable names for each registered model
Standardized Attribute & Variable Structure Lists

Attribute List
- attribute 1
- attribute 2
- attribute 3
- ...
- attribute n

Attribute Structure
- attribute name
- attribute type
- attribute data type
- attribute value

Variable List
- variable 1
- variable 2
- variable 3
- ...
- variable n

Variable Structure
- variable name
- variable data type
- variable size
- data classification
- variable values
- valid min
- valid max
- units
- grid system
- mask
- description
- is vector component
- position grid system
- data grid system
- actual min
- actual max

Model Data Assembled Into Standard Data Structures

write structure 2 standard format
Standardization is Feasible
( Summary from CCMC Workshop 2003 )

• BATRUS .Out to CDF conversion results promising
  – 1.5 second uncompressed CDF creation time
  – Resulting file size virtually unchanged

• OpenDx successfully imported CDF data using standard input module (only had to specify input file name)
  – Requires minimal initial development to correctly categorize imported data

• Working toward developing and implementing a flexible data format standardization software tool within the CCMC
Next Steps (from 2003 CCMC Workshop)

• Develop standard grid description scheme
• Implement HDF 5 conversion module
• Test BATRUS output conversion performance with HDF 5 data standard
• Compare CDF vs. HDF 5 performance
• Decide use of either CDF or HDF5 or both
• Develop standard “in-house” naming conventions for variables
Conversion Software

- generic attribute list (.h)
- model specific attribute list (.h)
- generic/default variable attribute list (.h)
- model specific variable attributes list (.h)
- Model Variable List
- Registered Variables List
  - CCMC name
  - native/alias
  - x: x_pos, x_pos
  - y: y_pos, y_pos

Conversion Routine

- main conversion routine
- main read driver
- read model a routine
- read model b routine
- main write driver
- write model a structure
- write structure .cdf
- write structure .hdf5

standard data files with common attributes and variable names for each registered model

CCMC Access/Interpolation Library

- open_cdf(cdf_name, 0); get_units(variable_name);
- interpolate_batsrus_cdf(variable, x, y, z, t, h);
- interpolate_ucla_qpcm_cdf(variable, x, y, z, t, h);
- close_cdf();
- get_attribute_float(attribute_name);
- get_attribute_char(attribute_name);
- Graf CDF Library:
  - get_float_data_path, start time, end time
  - get_interpolated_variable_name, start, y, z, t, h
  - get_attribute_variable(attribute_name, attribute_name);