INTRODUCTION TO MDSplus

by
Jeff Schachter

Presented at
General Atomics
San Diego, California

SEPTEMBER 22, 1999

DIII-D NATIONAL FUSION FACILITY SAN DIEGO

GENERAL ATOMICS

CLASS OUTLINE

Class 1: What is MDSplus and why should you care?

1. MDSplus — the philosophy
2. MDSplus at DIII–D
3. Basic concepts
4. MDSplus expressions
5. What you can do with MDSplus
6. Getting started

Class 2: How to use MDSplus

1. Overview
2. Examining tree structure
3. Reading data out of MDSplus
4. Writing data into MDSplus
5. Designing and building trees

DIII–D NATIONAL FUSION FACILITY SAN DIEGO, CALIFORNIA

GENERAL ATOMICS

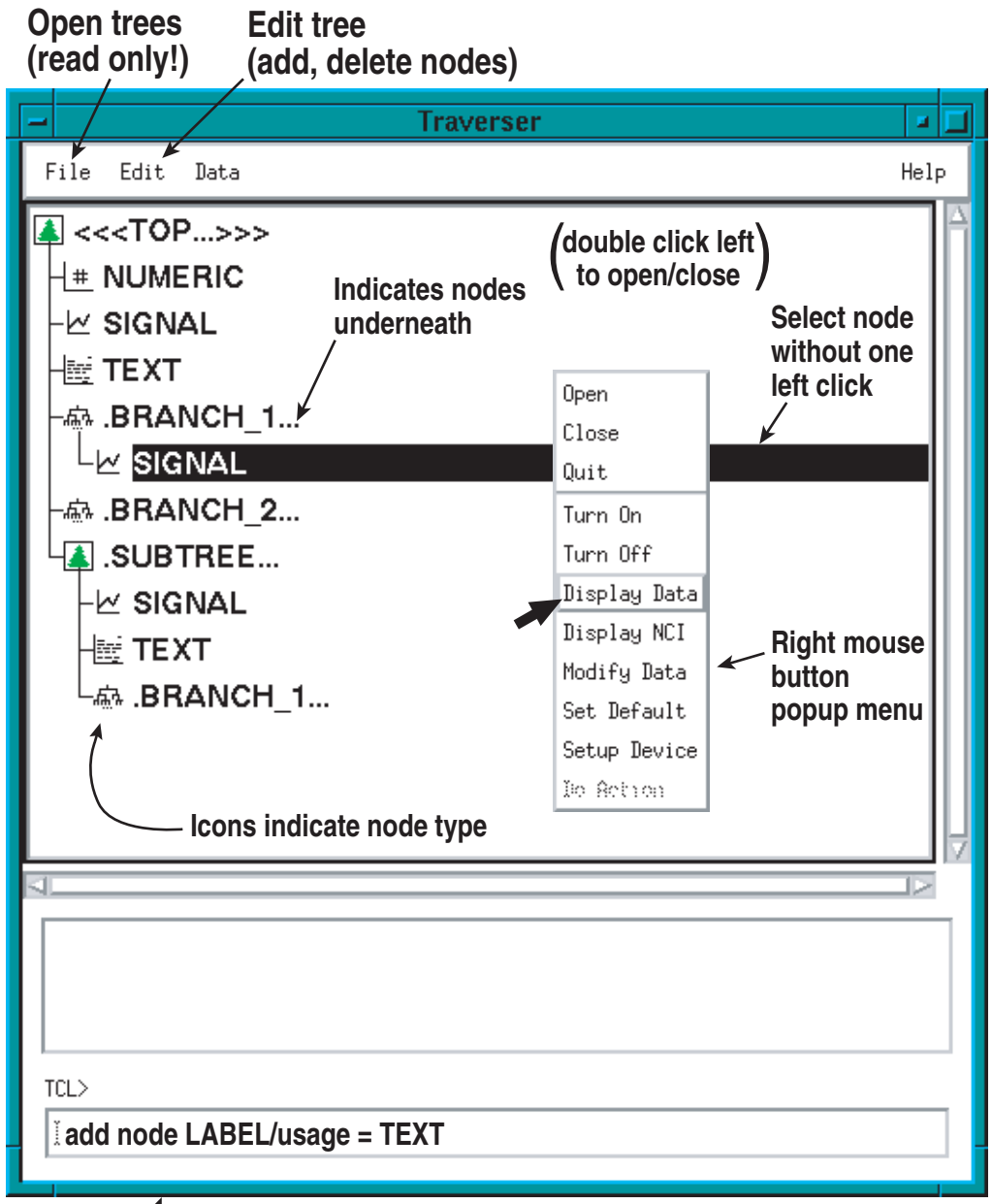
CLASS 2: HOW TO USE MDSplus

- This class presents the nuts and bolts of using MDSplus
- Overview
  - Examining tree (traverser)
  - Opening and closing trees
  - Reading data
  - Using MDSplus in existing tools (ReviewPlus)
  - Writing data to trees
  - Designing and building trees
  - Resources
- Use IDL for MDSplus function calls
  - If you would like C or Fortran information, please e-mail schacht@fusion.gat.com

DIII-D NATIONAL FUSION FACILITY SAN DIEGO, CALIFORNIA

GENERAL ATOMICS

TRAVERSER MDSplus GUI FOR EXAMINING TREES



USING TCL TO EXAMINE TREES

- TCL is "Tree Command Language": used for browsing and editing trees
- Like a file system
  - Current location in tree ("default")
  - Relative versus absolute paths
  - Set default, dir commands
- Decompile: examine contents of node (if not too large!)
- Show db: current tree and shot
- Can do TCL in traverser
- Useful for writing macros and generating large trees
- Demonstration

Start TCL from IDL	mdstcl
Open EFIT01 Model	set tree EFIT01/shot=–1/readonly
Directory of tree	dir
Change default	set def .RESULTS.AEQDSK
Examine node contents	decompile WMHD

OPENING AND CLOSING TREES

- Tree must be opened before data within can be accessed
- Opening a tree opens all its subtrees unless you say otherwise
  - Can open subtree directly
- Demonstration

  mdsopen, 'EFIT01',97979
- Trees do not need to be closed
  - Most recently opened tree is current context
  - Reopening an already open tree saves overhead
  - Maximum of eight trees open at once, 9th open automatically closes tree on bottom of stack

ASIDE: MDSplus STATUS CODES

- All MDSplus routines in IDL have optional STATUS=status keyword
- If status is odd, success. If status is even, failure
- Use in IDL:

```
mdsopen, tree, shot, /quiet, status=status
if (status) then begin
      [ . . . retrieve data . . . ]
endif else begin
      print, 'Error opening'+tree+strtrim(shot,2)+mdsgetmsg(status)
endelse
```

- /quiet keyword suppresses messages
- mdsgetmsg() function returns string explaining error message corresponding to status code

(3) READING DATA

- **Three steps**
  - Locate node of interest (tree, path, and shot)
  - Open tree
  - mdsvalue(path)

- **Example: Core Thomson Scattering T_e profile**

Open electrons tree, shot 97979

```
mdsopen, 'ELECTRONS', 97979
```

Ask for tag \TSTE_CORE

```
data = mdsvalue('\TSTE_CORE',/quiet,status=status)
```

Check status

```
print,status+' = '+mdsgetmsg(status)
```

NODE REFERENCE

- Node references can be
  - Absolute

    data = mdsvalue('\ELECTRONS::TOP.TS.BLESSED.CORE:TEMP')
  - Relative

    mdssetdefault,'\ELECTRONS::TOP.TS.BLESSED.CORE'

    data = mdsvalue('TEMP')
  - Tag

    data = mdsvalue('\TSTE_CORE')

- Recommend that you avoid relative node references for now
  - Absolute or tag references are easier to understand when examining code
  - No ambiguity or potential for mistakes

    mdssetdefault,'\ELECTRONS::TOP.TS.BLESSED.CORE'

    te = mdsvalue('TEMP')

    mdssetdefault,'-.DIVERTOR'

    t = mdsvalue('DIM_OF(TEMP)')

    → Gets T_e from core system, but timebase from divertor system

PASSING PARAMETERS TO MDSVALUE

- mdsvalue() function (and mdsput procedure) accept parameter arguments
- Primary argument to mdsvalue() is a string expression
- Use “$” within the string to represent a parameter passed in from IDL
- data = mdsvalue(‘SOME_FUNCTION($,$)’,arg1,arg2)
- Simple example:

```
arg1 = 2. * !PI * FINDGEN(101)/100
data = mdsvalue(‘SIN($)’,arg1)
```

- Will come up later in use of GETNCI() and especially mdsput

CACHING AND TDI VARIABLES

- Every time you retrieve a node reference, the tree is read
  - Can be painful if large amount of data
  - O/S does some caching
- Can use TDI variable to "cache" the data in memory
- Example:

No caching	Caching: use TDI variable
te = mdsvalue('\TSTE_CORE')	te = mdsvalue(' _s = \TSTE_CORE ')
t = mdsvalue('DIM_OF(\TSTE_CORE,0)')	t = mdsvalue('DIM_OF(_s,0)')
z = mdsvalue('DIM_OF(\TSTE_CORE,1)')	z = mdsvalue('DIM_OF(_s,1)')
u = mdsvalue('UNITS(\TSTE_CORE)')	u = mdsvalue('UNITS(_s)')
Four separate reads of node	_s is a TDI variable, now equal to the signal \TSTE_CORE

- All TDI variables start with underscore character "_"
  - Distinguish them from node references

USING TDI

- **Math operations**

  data = mdsvalue(' \TSTE_CORE / 1000. ')

- **Can have expressions containing multiple signals**
  - Example: |I/aB| from EFIT

    mdsopen, 'EFIT01',97979

    data = mdsvalue(' _s = ABS(\IPMEAS / \AMINOR / \BT0 / 1.e6)')

- **Be careful that signals are all the same shape**
  - In example, the EFIT signals all have the same timebase
  - MDSplus will not prevent you from doing the wrong thing

- **Signal format is lost: no more dimensions**

  print,mdsvalue('DECOMPILE('_s)')

  Build_With_Units([. . .],"A/m/T")

- **But units information retained**

  print,mdsvalue('UNITS(_s)')

  A/m/T

USING GETNCI

- **Arguments to GETNCI: GETNCI (node_reference, item_requested, [usage])**
- **node_reference**
  - Some node or set of nodes in a tree
  - Use "\\" if absolute path or tag ("\\TSTE_CORE", "\\TOP:NAMELIST")
  - Wildcards:

    * = all nodes at this level

    *** = all nodes at this level and below

    Example: all nodes one level down \TOP.RESULTS.AEQDSK in EFIT tree

    "\\EFIT01::TOP.RESULTS.AEQDSK:*"

    Example: All RESULTS nodes in EFIT tree

    "\\EFIT01::TOP.RESULTS***"
  - If using wildcards, return value must be same shape for each node
- **item_requested**
  - Full name, tag name (minimum path), length, relatives, data
  - Many different possibilities: see TDI help, GETNCI function
- **usage**
  - If specified, limit wildcard search to nodes of specified type
  - Signal, text, numeric, etc.

USING GETNCI — AN EXAMPLE

- Get names and labels of EFIT signals

- Get node ID numbers (NID) for further reference:

  nids = mdsvalue('GETNCI("\\EFIT01::TOP.RESULTS.*:*:LABEL","NID_NUMBER")')

- Get labels from each node

  labels = mdsvalue('GETNCI($,"RECORD")',nids)

- Get signal names

  signals = mdsvalue('GETNCI(GETNCI($,"PARENT),"MINPATH")',nids)

  - ➔ Inner GETNCI() gets parent NID numbers
  - ➔ Outer GETNCI() gets minimum path (tag) for each parent NID

GETTING PTDATA FROM MDSplus

- TDI function PTDATA() is interface between MDSplus and PTDATA
- PTDATA() returns signal containing data, timebase, and units
- Example: IP for shot 96021
  - data = mdsvalue('_s = PTDATA($,$)', "IP", 96021)
  - time = mdsvalue('DIM_OF(_s)')
  - units = mdsvalue('UNITS(_s)')
    - ➔ Note parameter passing and caching
- Can also get PTDATA headers
  - PTHEAD_IFIX(pointname,shot)
  - PTHEAD_RFIX(pointname,shot)
  - PTHEAD_ASCII(pointname,shot)
  - PTHEAD_INT16(pointname,shot)
  - PTHEAD_INT32(pointname,shot)
  - PTHEAD_REAL32(pointname,shot)
- Interpreting the headers is up to you!
  - See the DFI web pages http://lithos.ga.com/dfi/

(4) USING MDSplus IN EXISTING TOOLS

- MDSplus is already integrated into GA codes
  - ReviewPlus
  - EFITtools
  - GAprofiles
  - READA/G
  - IDL “get” routines
  - etc.

- Plotting pointnames in ReviewPlus and REVIEW
  - “pointname” — string designating signal of interest
  - MDSplus searched to determine tree and signal
  - If not MDSplus, then pseudopointname or PTDATA
    - ➔ ReviewPlus uses IDL to evaluate data combinations

- Can also use TDI directly in ReviewPlus
  - Specify tree name in addition to TDI expression
  - TDI expression evaluator used

(5) WRITING DATA TO TREES

- Practice on “scratch” trees first!
- Basic command: mdsput, node_reference, expression [. . . arguments]
- Example:
  - Open scratch tree

    mdsopen,’TEST’,10
  - Put signal into \TOP:SIGNAL

    expression = ‘BUILD_SIGNAL(BUILD_WITH_UNITS($,$),*,BUILD_WITH_UNTS($,$))’

    mdsput,’\TOP:SIGNAL’,expression,findgen(100)^2, ‘gizmos’,findgen(100), ‘ms’

    ➔ Note use of parameters (“$”) in expression: pass in IDL variables

MORE ON WRITING DATA

- MDSplus automatically logs username of person writing and data entered
  - See manually with TCL: dir/full
  - Retrieve with GETNCI (“OWNER_ID” and “TIME_INSERTED”)

- MDSplus does not keep track of previous contents of node
  - ➔ Be careful not to erase data

- No built-in mechanism for tracking revisions to data
  - Thomson scattering stores different revisions
  - But not a generic situation — solved specifically for them
  - Can use elements of their solution for other codes if needed

(6) DESIGNING AND BUILDING TREES

- Get your own data into MDSplus!
- 5 steps:
  - Identify information and data
  - Draw tree structure — how to organize? signal names?
  - Identify dependencies in data
  - Construct tree (using TCL)
  - Write procedure to load it
- Can be iterative process
- Use scratch tree to start
- Once finalized, will become part of D3D tree

CONSIDERATIONS WHEN WRITING DATA

- **Units! Units! Units**
  - Not mandatory, but very helpful
- **Take advantage of indirect referencing for timebases, etc.**
  - Saves space
  - Clearer organization
- **Calibration factors and raw data**
  - Use TDI expressions to allow changing factors without rewriting data
- **Cascading changes**
  - Changing one node affects others
  - Example: “blessed” Thomson revision: \ELECTRONS::TOP.TS:BLESSED_ID
- **Tradeoffs: performance, storage, clarity**
  - In the end — its up to you!

SIGNAL NAMES

- "Pointname" retrieval from MDSplus
  - User does not need to know tree
  - Instead MDSplus function searches all trees for tag
  - Signal tags must therefore be unique across entire DIII–D tree
  - MDSplus only requires unique tags within subtree

- Length limits
  - REVIEW: 10 characters
  - MDSplus node name: 12 characters
  - MDSplus tag: 23 characters

- To be backwards compatible, can only use 10 characters in tag

TCL FOR BUILDING TREES

- **ADD NODE node_reference [/USAGE=usage]**
  - node_reference
    - ★ Can be absolute or relative
    - ★ Use ".NODE" for STRUCTURE nodes
  - usage
    - ★ If node is not STRUCTURE, must specify
    - ★ SIGNAL, NUMERIC, TEXT are most common

- **DELETE NODE node_reference**
  - All nodes under node_reference also deleted
  - Will ask for confirmation

- **ADD TAG node_reference tag_name**
  - tag_name: do not need "\" character

- **WRITE**
  - Save your changes!

TCL TIPS AND TRICKS

- Use macros if adding same group of nodes repeatedly with changes
  - Macro is set of TCL commands grouped together
  - Ask Jeff if interested
- Can write TCL script files
  - Invoke with "@" sign
  - Can pass parameters (VMS syntax)
- Can send TCL commands from IDL
  - Uses mdstcl procedure: mdstcl,tcl_command_string
  - Very handy for writing programs to build trees

OTHER TCL CONSIDERATIONS

- **SET NODE command: determine automatic compression and read/write settings**
  - General form: SET NODE node_reference /qualifiers
  - Qualifiers:

/COMPRESS_ON_PUT	: compress data when written
/WRITE_ONCE	: cannot rewrite data
/MODEL_WRITE	: can write only into MODEL (shot –1) (or /NOMODEL_WRITE)
/SHOT_WRITE	: can write only into pulse tree (or /NOSHOT_WRITE)

- **Trees can be manually compressed and cleaned**
  - Compression: reduce space occupied by data
  - Cleaning: remove old data from files

SECURITY AND ACCESS CONTROL

- Right now, all users have all access to all trees
  - BE CAREFUL!
- Will soon move to system where write access is restricted
  - Will be done by group membership
  - Example: all spectroscopists will have write access to SPECTROSCOPY tree
- Access is granted at the subtree level
  - Cannot control access to parts of the tree
- Offsite access will start as read only
  - Can grant write access to specific users from specific sites

(7) RESOURCES

- DIII–D Data Analysis Group website: http://fusion.gat.com/comp/analysis
  - MDSplus at DIII–D
  - Analysis code documentation
  - Signals documentation
  - Links to MIT site, online MDSplus help
- E-mail list? Newsgroup?
  - Wait to see demand
- Ask Jeff
  - X4168, 13/413
  - Always happy to answer questions

SUMMARY

- MDSplus is ready for use!
- Little added effort will return many more benefits
  - Direct access to data
  - Know exactly what you get
  - Retrieve more than just the numbers
- Get your data into MDSplus
  - Once in, everyone can see it
  - More efficient for everyone if you do the design work
  - Jeff will help and work on the “big” sets